ifddt Benutzerhandbuch

Stefan Weideneder
Copyright 1999−2009 SEB Weideneder, Munich

Inhaltsverzeichnis

1 Installationsanleitung 1
1.1 Produktverzeichnis anlegen 1
1.2 Umgebungsvariablen 1
1.3 Kopieren und Entpacken der Software 1
1.4 Installation der Software 2
1.5 Demo−Versionen 3

2 Funktionsweise 5
2.1 Funktionsbeschreibung 5
2.2 Extent−Größen 6
2.3 Reihenfolge der Tabellenanlage 6
2.4 Arbeiten mit der storage−Datei 6
2.5 Fehlerhafte storage−Datei 9

3 Umgebungsvariablen 11
3.1 DDTINVIEW 12
3.2 DDTDEB 13
3.3 DDTST 14
3.4 DDTCOMP 15
3.5 MINHPLROWS 16
3.6 MAX_FILE_SIZE 17
3.7 IFDDT_CONN_RETRY und IFDDT_CONN_TIME 18
3.8 DBDELIMITER, DBMONEY und DELIMIDENT 19

4 Bedienungsanleitung 21
4.1 Datenbank−Schema erstellen 21
4.2 Serverspezifische Tabelleneigenschaften ausgeben (−s) 21
4.3 Größeninformationen auswerten 22
4.4 Standard−Constraint−Namen unterdrücken (−c) 23
4.5 Not−Null−Constraint−Namen unterdrücken (−C) 23
4.6 Erzeugen einer Standard storage Datei (−S) 23
4.7 Extent−Multiplikator festlegen (−e/−E factor) 24
4.8 Zielversion festlegen (−u) 24
4.9 Entladen einer Datenbank (−N) 25
4.9.1 Restart Option (−w) 26
4.9.2 Entladedateien komprimieren (−Z level) 27
4.9.3 Umgebungsvariable DBMONEY 27
4.9.4 Alphanumerische Werte konvertieren (−k datei) 28
4.10 Verzögerte Indexanlage (−i und −O) 29

Inhaltsverzeichnis

4 Bedienungsanleitung

4.11 Indizes reorganisieren (−I dateiname) 30
4.12 Datenbanken kopieren (−n name) 31
4.12.1 Clustered Indizes 32
4.13 Tabellen reorganisieren (−n) 32
4.14 Umgebungsvariable INFORMIXSERVER setzen 32
4.15 Kopieren einer Datenbank innerhalb eines Servers 33
4.16 Tabellen selektieren (−|+t) 34
4.16.1 Tabellenrechte nicht unterdrücken (−p all) 34
4.16.2 Metazeichen in Tabellennamen 35
4.16.3 Zugehörige Views entladen (−v) 35
4.16.4 Tabelle vor der Anlage löschen (−r/−R) 35
4.16.5 Stored Procedures ausgeben (−f) 36
4.17 Tabellen kopieren 36
4.18 Transformationen (−F datei) 36
4.19 Allgemeine Optionen 37
4.20 Benutzernamen unterdrücken (−a) 37
4.21 Bestimmte Benutzertabellen ausgeben (−o owner) 38
4.22 Views, Trigger, Procedures und Synonyme auslagern (−b datei) 38
4.23 Schnelle Indexanlage (−i) 38
4.24 Indizes auslagern (−i datei) 39
4.25 Indizes reorganisieren (−I datei) 40
4.26 Constraints auslagern (−B datei) 40
4.27 Implizite Indizes umgehen(−G) 41
4.28 Indizes prüfen(−A datei) 42
4.29 Zusatzoptionen bei Storage Dateien 43
4.29.1 DbSpace Definitionen festlegen (−D dbspacelist) 43
4.29.2 Extent Berechnung anhand monatlicher Wachstumswerte (−M month) 44
4.30 DbSpace Mapping (−m) 44
4.31 Tabellenliste generieren (−y datei) 44
4.32 Tabellensuche API (−z) 45
4.33 Update Statistics (−U) 46
4.34 Cache Optimierung (−O) 46
4.35 Datenbanksperre beim Entladen vermeiden (−l) 46
4.36 High Performance Loader einsetzen (−H datenbank) 47
4.37 Optionen zur Parallelverarbeitung (−g) 49
4.38 Datenbanknamen spezifizieren (−g dbs=name) 49
4.39 Autoselect der Parallelverarbeitung (−g ppr) 49

Inhaltsverzeichnis

4 Bedienungsanleitung

4.40 Anzahl der parallelen Prozesse festlegen (−g anzahl_prozesse) 50
4.41 PDQ−Priorität festlegen (−g pdq) 50
4.42 Ausgabe für Oracle erzeugen (−g ora) 51
4.43 inf_storage Datei für BaaN Umgebungen erzeugen 51

5 Administrative Anforderungen 53

5.1 Einrichten einer Server−Server−Verbindung 53
5.2 ONCONFIG−Konfigurationsparameter 53
5.3 Beschleunigung des Ziel−Servers 54
5.4 Voraussetzungen der Web−Server−Konfiguration 54
5.5 Festplatten−Layout 55

6 Bekannte Bugs 57

1 Installationsanleitung

Bitte führen Sie die Installation des Softwareprodukts **ifddt** entsprechend dieser Installationsanleitung durch.

1.1 Produktverzeichnis anlegen

Melden Sie sich zunächst unter der Benutzerkennung **root** an und richten Sie für die **iidsp** Produkte ein eigenständiges Unterverzeichnis ein, sofern Sie dies nicht bereits für andere **iidsp** Programme durchgeführt haben.

```
$ /bin/su
$ mkdir /opt/informix/iidsp
```

1.2 Umgebungsvariablen

Setzen Sie anschließend die Umgebungsvariable **IIDSPDIR** auf den absoluten Pfadnamen des Produktverzeichnisses. Unter Verwendung der **C−Shell**setzen Sie Umgebungsvariablen mit folgendem Befehl:

```
% setenv IIDSPDIR /opt/informix/iidsp
```

Unter Verwendung der **Bourne**−Shell oder **Korn−Shell** setzen Sie Umgebungsvariablen folgendermaßen:

```
$ IIDSPDIR=/opt/informix/iidsp
$ export IIDSPDIR
```

In den nachfolgenden Beispielen arbeiten wir durchgehend mit der Korn−Shell. Sofern Sie die C−Shell oder eine andere Shell anwenden, müssen Sie die Beispiele jeweils anpassen.

Wechseln Sie anschließend in das Produktverzeichnis:

```
$ cd $IIDSPDIR
```

1.3 Kopieren und Entpacken der Software

Kopieren Sie nun die Installationsdateien vom Installationsmedium in das Produktverzeichnis

und entpacken Sie die Installationsdatei:

```
$ cp /mnt/ifddt.tar $IIDSPDIR
$ tar xf $IIDSPDIR/ifddt.tar
```

Anschließend sollte das Produktverzeichnis folgende Unterverzeichnisse aufweisen:

.	
bin	Enthält die ausführbaren Programme
doc	Enthält die Dokumentation
etc	Enthält die Installationsdateien
msg	Enthält Nachrichtentexte aller Art
release	Enthält Freigabemitteilungen

1.4 Installation der Software

Zur weiteren Installation der **ifddt** Software sollten Sie nun die **Seriennummer** und den **Activation Key** Ihres Software−Lizenzvertrages bereithalten. Beide Werte befinden sich auf dem Aufkleber auf der Rückseite Ihres Lizenzvertrages und werden während der nachfolgenden Installationsroutine abgefragt.
Starten Sie das Installationsprogramm wie folgt:

```
$ ./installifddt
```

Kurz darauf erscheint die Meldung:

```
Installation Script

This installation procedure must be run by root (super−user)
It will change the owner, group, and mode of all files of
this
package in this directory. There must be a user "informix"
and a
group "informix" known to the system.

Press RETURN to continue,
or the interrupt key (usually CTRL−C or DEL) to abort
```

Drücken Sie an dieser Stelle die RETURN−Taste und es folgt die Aufforderung zur Eingabe Ihrer **Seriennummer**:

```
Enter your serial number (for example, IFS#R999999) >
```

Geben Sie hier Ihre Seriennummer ein. Buchstaben sollten dabei stets groß geschrieben werden. Bestätigen Sie Ihre Eingabe mit der RETURN−Taste, und es erscheint die Aufforderung zur Eingabe des **Aktivierungsschlüssels**:

```
Enter your serial number KEY (uppercase letters only) >
```

Schließen Sie die Eingabe erneut mit der RETURN−Taste ab. Daraufhin erscheint eine kurze Meldung, in der Sie die Lizenzbedingungen mit Hilfe der RETURN−Taste bestätigen müssen. Im Anschluß daran werden die verschiedenen Unterverzeichnisse angezeigt, deren Rechte auf die Benutzerkennung **informix** gesetzt werden. Zudem werden die Zugriffsbeschränkungen der Programme aufgehoben. Sobald die Meldung:

```
Installation complete.
```

auf Ihrem Bildschirm erscheint, ist die Installationsroutine abgeschlossen.

Testen Sie jetzt das Programm, indem Sie den Pfad um das Produktverzeichnis *$IIDSPDIR/bin* erweitern und **ifddt** zum ersten Mal starten:

```
$ IIDSPDIR=/opt/informix/iidsp
$ PATH=$PATH:$IIDSPDIR/bin
$ ifddt −V
```

1.5 Demo−Versionen

Demo−Versionen haben ausgehend vom Zeitpunkt der Installation eine 30−tägige Gültigkeit. Demo−Versionen sind mit dem vollen Leistungsumfang ausgestattet, dürfen aber nicht zu kommerziellen Zwecken eingesetzt werden.

Nach Ablauf der 30−Tage Frist erlischt automatisch der Lizenzvertrag und die Software darf nicht mehr angewendet werden.

2 Funktionsweise

ifddt ist ein Programm, mit dessen Hilfe Datenbanken oder Tabellen kopiert oder entladen werden können und eignet sich somit optimal zur Reorganisation sowie zur Erstellung von Kopien von Datenbanken oder Tabellen. Ein wesentliches Merkmal ist, dass **ifddt** nur die zum Kopieren oder Laden notwendigen Befehle erzeugt, diese aber nicht sofort ausführt. Die Ausführung wird letztlich mittels des Programms »dbaccess« realisiert. Somit lassen sich vor einer anstehenden Reorganisation die bevorstehenden Schritte überwachen und sind leichter zu koordinieren.
ifddt verwendet die native Informix Schnittstelle für den Zugriff auf die IBM Informix Systemtabellen. Zur Ermittlung der realen Extent−Größen ist es nicht nur notwendig, auf den Systemkatalog der zu bearbeitenden Datenbank zuzugreifen, es müssen auch einige Verwaltungstabellen der sysmaster Datenbank gelesen werden. Je mehr Tabellen innerhalb einer Server−Instanz vorhanden sind und somit über die sysmaster Datenbank verwaltet werden, desto länger dauert auch der erste Verarbeitungsschritt des Programms **ifddt**.
Im ersten Schritt ermittelt **ifddt** die spezifischen Größen der Partition−Informationen der Datenbanktabellen und hinterlegt diese temporär in einem dynamischen Speicher. Intern werden die Partition−Informationen später in einem eigenen Hash−Join−Verfahren mit den Tabelleninformationen des Systemkatalogs verknüpft. Anschließend werden alle Tabellen des Systemkatalogs gelesen und ausgewertet.
Um sicherstellen zu können, dass jede Tabelle nur einmal gelesen wird, entstehen einige temporäre Dateien, welche im Verzeichnis ***/tmp*** bzw. **$DBTEMP** abgelegt werden. In diesen temporären Dateien befinden sich in nicht sortierter Reihenfolge die SQL−Befehle zur Anlage von Tabellen, Indizes und der Ladeanweisungen. Die Gesamtgröße aller temporären Dateien entspricht der Gesamtgröße der letztendlich erstellten SQL−Schema Datei. Erst im letzten Schritt werden die Anweisungen aus den verschiedenen temporären Dateien sortiert gelesen und in die Ausgabe übertragen. Aufgrund dieser Sortierung kann eine gewisse Zeit verstreichen, bis die ersten Tabellenbefehle in der Ausgabe erscheinen.

2.1 Funktionsbeschreibung

ifddt erzeugt eine Schema−Datei zur Anlage einer Datenbank, oder aber zur Anlage bestimmter Tabellen einer Datenbank − optional auch die zugehörigen Lade−Anweisungen. In der weiteren Beschreibung wird grundsätzlich unterschieden zwischen den Schemata zum Erstellen ganzer Datenbanken und denen zur Erstellung einzelner Tabellen.
Sollen Datenbanken oder einzelne Tabellen kopiert werden, so kann dies entweder direkt, oder über eine Zwischenablage geschehen. Das Erstellen einer Zwischenablage geschieht mit Hilfe so genannter unload−Dateien und wird daher im Folgenden als »Entladen« bezeichnet.

2.2 Extent−Größen

Die bei der Tabellenanlage einzusetzenden Extent−Größen hängen von folgenden Faktoren ab:
ifddt ermittelt die aktuelle Tabellengröße aus den Partition−Informationen der sysmaster−Datenbank. Darüber hinaus kann die ausgegebene Extent−Größe mit Hilfe der Extent Option in der Kommandozeile, oder mit Hilfe einer Definition in der **storage−Datei** verändert werden. Während sich die Extent Option in der Kommandozeile auf alle in dem Verarbeitungsprozeß angesprochenen Tabellen bezieht, kann mittels der **storage−Datei** für jede Tabelle eine individuelle Extent−Größe vereinbart werden. Mehr zur Anwendung der **storage−Datei** erfahren Sie unter »Arbeiten mit der **storage−Datei**«.

2.3 Reihenfolge der Tabellenanlage

ifddt erstellt immer eine Schema−Datei, die standardmäßig auf dem Standard−Ausgabekanal ausgegeben wird. Die Schema−Datei enthält alle SQL−Befehle zur Anlage der entsprechenden Tabellen. Die Befehle erscheinen in der Ausgabe stets in einer festen Reihenfolge: Begonnen wird mit den größten Tabellen, um diesen bei der Anlage den freien Platz in den DbSpaces weitgehend garantieren zu können. Würde abschließend der freie Platz für die kleinsten Tabellen nicht mehr ausreichen, so würde noch vor dem Laden der ersten Tabelle eine Fehlermeldung erscheinen.

2.4 Arbeiten mit der storage−Datei

Sollen bei einzelnen Tabellen einer bestehenden Datenbank gezielt Änderungen im server−spezifischen Bereich vorgenommen werden, so kann dies über die Zuhilfenahme einer **storage−Datei** geschehen. Der Aufbau der **storage−Datei** erlaubt es, Direktiven für eine oder mehrere Tabellen verschiedener Datenbanken vorzugeben. Neben den Direktiven für Tabellen sind auch Direktiven für Indizes zulässig. Erfolgt die Angabe einer Index−Direktive, so geht **ifddt** davon aus, dass es sich bei diesem Index um einen detached Index handelt, der möglicherweise in einem separaten DbSpace gespeichert wird. Die Angaben der **storage−Datei** umfassen neben der Datenbank und der Tabellen− bzw. Indexangabe folgende Direktiven:

- die Sperrebene (Page− oder Row−Lock)
- den Ziel−DbSpace für eine Tabelle/Index bzw. den Fragmentierungsausdruck
- die Anzahl der Datensätze, um die eine Tabelle monatlich wächst bzw. der Index−Füllfaktor

- die First− und Next−Extent−Größen
- die Angabe, ob eine Tabelle resistent im Cache verweilen soll

Alle Einträge innerhalb der **storage−Datei** müssen Sie durch einen Doppelpunkt voneinander trennen. Die einzelnen Tabellen− bzw. Indexangaben beginnen stets auf einer neuen Zeile. Jeder neue Eintrag in der **storage−Datei** beginnt mit dem Namen der Datenbank, gefolgt von einem Doppelpunkt (:) und dem Namen der Tabelle bzw. dem Namen des Indexes. Handelt es sich um eine Tabellenbeschreibung, so muß nach dem Tabellennamen der Eintrag **»T«** folgen; handelt es sich um einen Index, so muß der Eintrag **»I«** folgen.
Bei Indexbeschreibungen sind nur zwei Direktiven möglich. Diese befinden sich in der achten und neunten Spalte und beschreiben den Füllfaktor bei der Index−Anlage sowie den Ziel−DbSpace bzw. den Fragmentierungsausdruck.
Der vollständige Aufbau bei einer Tabellen−Direktive sieht folgendermaßen aus:

datenbank	Diese Spalte enthält den Namen der Datenbank, für die der nachfolgende Eintrag gilt.
tabelle	In dieser Spalte wird der Tabellen bzw. der Indexname eingetragen.
typ	Die Angabe **T** in dieser Spalte bedeutet, dass es sich um einen Tabelleneintrag handelt. Der Wert **I** bedeutet, dass es sich um einen B−Tree Index handelt.
cache	Enthält die Spalte *cache* den Buchstaben **R**, so soll die Tabelle **resistent** im Hauptspeicher geladen werden, sobald der Datenbankserver gestartet wird. Diese Angabe ist optional und nur im Zusammenhang mit dem *»Central Monitoring System« aktiv*.
sperrebene	Die Spalte *sperrebene* kann entweder ein **P** für Page−Lock oder den Buchstaben **R** für Row−Lock enthalten. Bei Indexangaben ist diese Spalte ohne Bedeutung.
firstextent	Die Spalte firstextent können Sie mit der ***initialen Extent−Größe*** der Tabelle versehen. Ist die Angabe jedoch kleiner als die tatsächliche, momentane Tabellengröße, so wird diese Angabe ignoriert. Die Extent−Größen müssen Sie in der Einheit Kilobyte angeben. Sofern Sie neben der **storage−Datei** auch die Extent−Option in der Kommandozeile angegeben haben, so wird die in der First−Extent Spalte definierte Größe mit dem Extent−Faktor der Kommandozeile multipliziert.
nextextent	

	Die Spalte *nextextent* belegen Sie mit der Größe der weiteren Extents. Falls Sie diese Spalte leer lassen, so verwendet **ifddt** stets ein Drittel der First−Extent Größe.
wachstum	Die *wachstum* Spalte kann optional mit der Anzahl der Datensätze belegt werden, um die eine Tabelle durchschnittlich je Monat anwächst. Diese Information wird seitens **ifddt** ausgewertet, wenn Sie in der Kommandozeile die Option **−M monate** spezifizieren. Aufgrund der jeweiligen Satzbreite der Tabellen ermittelt **ifddt** die zusätzliche Größe der Tabellen nach Ablauf der in der Kommandozeile festgelegten Monate, wobei von einem linearen Wachstum ausgegangen wird. Die in der firstextent Spalte vereinbarte Größe wird dann um den Wachstumswert erhöht.
space	Die *space* Spalte dient dazu, einen Ziel−DbSpace für eine Tabelle zu definieren. Sofern Sie in dieser Spalte einen DbSpace−Namen eintragen, wird die Tabelle bzw. der Index in diesem DbSpace angelegt. Bei dem DbSpace−Namen muß es sich nicht um einen real existierenden DbSpace im aktuellen Server−System handeln; entscheidend ist, dass der DbSpace in dem Zielsystem existiert, in dem das erzeugte SQL−Schema angewendet wird. Neben einem DbSpace−Namen kann die DbSpace Spalte auch einen Fragmentierungsausdruck enthalten, jedoch muß der vollständige Ausdruck in einer Zeile untergebracht werden. Die bei einer Tabellendefinition vorgenommenen Extent−Angaben beziehen sich in diesem Fall auf jedes Daten−Fragment. Bleibt die *space* Spalte für eine Tabelle leer, so ermittelt **ifddt** einen passenden DbSpace für diese Tabelle: Wird zusätzlich die Option **−D dbspacelist** in der Kommandozeile angegeben, so verwendet **ifddt** einen DbSpace aus der spezifizierten DbSpace Definition, wobei eine gleichmäßige Nutzung der definierten DbSpaces angestrebt wird. Ist jedoch keiner der definierten DbSpaces hinreichend groß, um diese Tabelle aufzunehmen, so wird die Tabelle über die definierten DbSpaces im Round Robin Modus fragmentiert. Die in der DbSpace−Option der Kommandozeile definierten DbSpaces müssen im aktuellen Server−System nicht enthalten sein. Daher ist bei der DbSpace−Option neben dem DbSpace−Namen auch die DbSpace−Größe in der Einheit Kilobyte anzugeben. Existiert ein DbSpace gleichen Namens im aktuellen

	Server−System, so wird die in der DbSpace−Option angegebene DbSpace−Größe als maßgebliche Größe verstanden. Fehlt die Angabe der Option **−D** in der Kommandozeile, so wird die Tabelle in dem DbSpace angelegt, in dem auch die Datenbank angelegt wurde.

Die **storage−Datei** wird seitens **ifddt** stets um fehlende Einträge ergänzt und mit aktuellen Werten abgeglichen. Das heißt, dass fehlende Tabellen, fehlende Next−Extent− oder DbSpace−Angaben nachgetragen und neu berechnete Extent−Größen modifiziert werden. Daher ist es auch notwendig, dass Sie als aufrufender Benutzer die Rechte haben müssen, die **storage−Datei** zu modifizieren. Die alte **storage−Datei** wird jedoch stets unter dem gleichen Pfadnamen mit der Endung *.bak* kopiert. Die Tatsache, dass **ifddt** die **storage−Datei** um fehlende Einträge ergänzt, hilft Ihnen bei der Erstellung einer Standard **storage−Datei**. Die generierte **storage−Datei** können Sie in einem zweiten Arbeitsschritt individuell an Ihre Bedürfnisse anpassen.

2.5 Fehlerhafte storage−Datei

Falls **ifddt** innerhalb der **storage−Datei** einen Fehler erkennt, so wird die weitere Verarbeitung abgebrochen und es erscheint eine Fehlerausgabe der Form:

```
$ ifddt −d stores7 −S /tmp/weidi73.ddt > /dev/null
storage file '/tmp/weidi73.ddt' error −3 in line 2 !
```

Wird ein Fehler erkannt, so erstellt **ifddt** weder eine neue **storage−Datei**, noch ein Schema.

Die negativen Fehlernummern während der Fehlerprüfung der **storage−Datei** bedeuten im einzelnen:

Fehler−Nr	Bedeutung
−1	Es fehlt der Datenbankname
−2	Es fehlt der Tabellenname
−3	Die Tabellen/Index−Info fehlt
−4	Der Cache Eintrag ist falsch
−5	Cache bei Index gesetzt
−6	Ungültiger Sperrmodus

−7	Es fehlt die First Extent Größe
−8	Es fehlt die Next Extent Größe
−9	Es fehlt der Wachstums Faktor
−21	Es fehlt das ; Endezeichen
−22	Der DbSpace Name ist zu lang
−23	Fehler im Fragmentausdruck

3 Umgebungsvariablen

Mit Hilfe der folgenden Umgebungsvariablen lässt sich das Verhalten des **ifddt** beeinflussen:

- DDTINVIEW
- DDTDEB
- DDTST
- DDTCOMP
- MINHPLROWS
- MAX_FILE_SIZE
- IFDDT_CONN_RETRY
- IFDDT_CONN_TIME
- DBDELIMITER
- DBMONEY
- DELIMIDENT
- ORA_IFDDTPATH
- ORA_USER
- ORA_PASSWD
- ORA_LOGPATH
- ORA_FRACTIONS
- BAANFILES

Die Variablen werden nachfolgend im einzelnen hinsichtlich der möglichen Werte und ihrer Bedeutung beschrieben. Lesen Sie bitte jeweils im Kapitel 4 mehr zu den jeweiligen Optionen. Diese Beschreibung soll lediglich eine grobe Übersicht über die verschiedenen Variablen darstellen.

In den nachfolgenden Beispielen beschreiben wir das Setzen der Umgebungsvariablen für die Bourne−Shell oder ihrer Derivate wie Korn−Shell oder der bash. Sofern Sie eine andere Shell zur Kommandoeingabe nutzen, setzen Sie die Umgebungsvariablen dort bitte entsprechend.

3.1 DDTINVIEW

Diese Umgebungsvariable aktiviert eine spezielle Funktion, bei der Views analysiert werden. Wird die Umgebungsvariable auf den Wert 1 gesetzt, so werden lediglich Eigentümernamen innerhalb des Views in der Kombination mit der Option **−a** entfernt. Wird die Umgebungsvariable auf den Wert 2 gesetzt, so werden die Views hinsichtlich externer Tabellenreferenzen aufglöst. Anstatt die Views regulär zu übernehmen, werden externe Tabellenreferenzen wie interne Tabellen behandelt, d.h. die Daten der Tabelle werden je nach Option entweder mit kopiert oder ent− und geladen. Entsprechend findet sich dann auch im Schema ein entsprechender Befehl zur Tabellenanlage.

```
$ DDTINVIEW=2
$ export DDTINVIEW
$ ifddt −d checkdata −c −C −s −i −z −y checkrows.dat −n newdwh −R −g 8 > newdwh.sql
```

3.2 DDTDEB

ifddt enthält eine Option zum einfachen Verfolgen des internen Programmablaufs. Wird diese Debugging Funktion mittels der Umgebungsvariable DDTDEB aktiviert, so erscheint in der Fehlerausgabe des ifddt eine Auflistung der internen Funktionen. Die eigentlich zu generierende Ausgabe ist in diesem Fall nicht mehr sinnvoll zu verwenden. Sofern das Programm jedoch stets in einen Fehler läuft oder gar die Aufgabe abbricht, ist dies in der Regel eine sehr schnelle Option, um den Support bei der Fehlersuche zu unterstützen. Folgen Sie zum Setzen der Variable daher den Anweisungen Ihres Support−Teams.

```
$ DDTDEB=2
$ export DDTDEB
$ ifddt −d checkdata −c −C −s −i > /dev/null 2> debug.log
```

3.3 DDTST

Standardmäßig werden seitens **ifddt** nur Tabellen ausgegeben, welche die Datenbankadministratoren angelegt haben. Systemtabellen können ausgegegen werden, sofern die Umgebungsvariable vor dem Aufruf auf irgendeinen Wert gesetzt wird. Prinzipiell ist der Aufbau der Systemtabellen zwar nur selten von Interesse, jedoch lassen sich auf diese Weise schnell Unterschiede zwischen verschiedenen Versionen erkennen.

```
$ DDTST=1
$ export DDTST
$ ifddt −d checkdata −c −C −s −i −t "sys*" > checkdata.sys
```

3.4 DDTCOMP

Beim Entladen können Tabellen generell auch komprimiert entladen werden, indem die Option **−Z** eingesetzt wird. Die dazu verwendeten Programme sind *gzip*, *compress* oder *zip* mit den jeweiligen Dekompressionsprogrammen. Sofern das ihrerseits gewünschte Programm nicht vorhanden ist, können Sie ein eigenes Programm wählen. Tragen Sie das Kompressions− und ein Dekompressionsprogramm getrennt durch ein Semikolon in die Umgebungsvariable **DDTCOMP** ein. Beide Programme müssen Filterprogramme sein, d.h. sie müssen ohne Angabe eines Dateinamens von der Standardeingabe lesen und auf die Standardausgabe schreiben.

```
$ DDTCOMP="bzip2;bunzip2"
$ export DDTCOMP
$ ifddt −d checkdata −N neu.exp −H neu −c −C −s −i −Z level=0
```

3.5 MINHPLROWS

Beim Entladen mittels High Performance Loader werden Tabellen mit weniger als 10.000 Datensätzen mittels dem dbaccess Befehl **load** geladen. Der Aufwand für den Aufruf des High Performance Loaders ist in diesen Fällen oftmals zu groß. Sie können diesen Schwellwert mittels der Umgebungsvariable **MINHPLROWS** nach unten oder oben setzen.

```
$ MIN_HPL_ROWS=80000
$ export MIN_HPL_ROWS
$ ifddt −d checkdata −N neu.exp −H neu −c −C −s −i
```

3.6 MAX_FILE_SIZE

Beim Entladen von Daten sieht ifddt eine maximale Dateigröße von 2GB vor, sofern die Daten in reguläre Dateien geschrieben werden. Wenn Sie größere Dateien wünschen und ein entsprechendes Dateisystem vorliegt, so können Sie die Umgebungsvariable **MAX_FILE_SIZE** auf einen anderen Wert setzen. Der Wert muß dabei in Bytes angegeben werden. Ist die zu entladende Datei größer als MAX_FILE_SIZE, so werden mehrere Entladedateien erzeugt.

3.7 IFDDT_CONN_RETRY und IFDDT_CONN_TIME

Diese Umgebungsvariablen beeinflussen den Verbindungsaufbau des **ifddtcp**. Ist der Dämonprozess zum Zeitpunkt des Verbindungsaufbaus noch nicht gestartet, so wiederholt der **ifddtcp** Prozess den Vorgang sooft, wie Sie in der Umgebungsvariablen **IFDDT_CONN_RETRY** eingetragen haben. Die Zeitspanne zwischen den Versuchen können Sie in Sekunden mittels der Umgebungsvariable **IFDDT_CONN_TIME** definieren. Setzen Sie diese Variablen unmittelbar vor dem Start der Datenübernahme.

```
$ IFDDT_CONN_RETRY=3
$ IFDDT_CONN_TIME=10
$ export IFDDT_CONN_RETRY IFDDT_CONN_TIME
$ ifddt −d checkdata −nb neu −s −c −C −i −W  | dbaccess −
```

3.8 DBDELIMITER, DBMONEY und DELIMIDENT

Bei diesen Variablen handelt es sich um reguläre Informix Dynamic Server Umgebungsvariablen, welche beim Entladen und Laden von Daten die Ausgabe beeinflussen. Die Wirkungsweise der Variablen wird seitens des **ifddt** entsprechend unterstützt. Wird DELIMIDENT eingesetzt, so ändern sich unter anderem die Namen der Entladedateien; diese werden in diesem Fall entsprechend durchnumeriert.

4 Bedienungsanleitung

Mit Hilfe von **ifddt** können Sie entweder

- ein Datenbankschema erstellen,
- ein Tabellenschema erstellen,
- ein Skript zum Kopieren ganzer Datenbanken anlegen,
- ein Skript zum Kopieren einzelner Tabelle erzeugen,
- Datenbanken entladen und ein Skript zum Laden generieren,
- Tabellen entladen und ein Skript zum Laden dieser Tabellen erstellen,
- Indizes einer ganzen Datenbank oder einzelner Tabellen reorganisieren.

Je nachdem, welche der Hauptfunktionen Sie nutzen möchten, müssen Sie das Programm **ifddt** mit unterschiedlichen Optionen starten. Die Anwendung dieser Optionen wird im Laufe dieses Kapitels erklärt.

4.1 Datenbank−Schema erstellen

Zum Erstellen eines Datenbankschemas genügt es, **ifddt** mit der Option **−d** gefolgt von einem Datenbanknamen zu starten:

```
$ ifddt −d stores7
```

Standardmäßig wird das Schema zur Erstellung der ganzen Datenbank ausgegeben. Bei den Constraint−Definitionen werden standardmäßig deren Namen ausgegeben; auch dann, wenn es sich um die vom System selbst generierten Namen handelt. Ohne Angabe einer weiteren Option fehlen in der Ausgabe alle server−spezifischen Tabelleneigenschaften. Diese können Sie mit Hilfe der Option **−s** aktivieren:

```
$ ifddt −d stores7 −s
```

4.2 Serverspezifische Tabelleneigenschaften ausgeben (−s)

Zu den serverspezifischen Tabelleneigenschaften gehören

- die First− und Next Extent Size Definitionen
- die Ziel−DbSpaces der Tabellen

- der Sperrmodus der Tabellen
- die Fragmentierungsausdrücke für die Tabellen

Diese Tabelleneigenschaften werden seitens **ifddt** ausgegeben, sofern Sie die Option **−s** auf der Kommandozeile angeben.

Wenn Sie die server−spezifischen Ausgaben aktivieren, so ermittelt **ifddt** nicht nur die Größen der Benutzertabellen, sondern korrigiert auch die Größen der Systemtabellen. Da sich jedoch die First Extent Größe der Systemtabellen nicht beeinflussen lässt, korrigiert **ifddt** deren Next Extent Größen, so dass die Systemtabellen nach der Anlage aller Tabellen nur zwei Extents aufweisen. Die Option **−s** veranlasst **ifddt** auch dazu, die Tabellen in absteigender Reihenfolge ihrer Größe auszugeben. Ohne diese Option werden die Tabellen entsprechend des Zeitpunktes ihrer Anlage ausgegeben.

4.3 Größeninformationen auswerten

ifddt erzeugt bei Angabe der Optionen **−s** oder **−S** am Ende des Schemas eine Größeninformation. Diese Größeninformation befindet sich innerhalb der Kommentarzeichen *{* und *}* und muss daher nicht manuell entfernt werden.
Die Größeninformation zeigt Ihnen, wieviel Platz für die Tabellen innerhalb der verschiedenen DbSpaces benötigt wird. Dabei werden auch die Namen der DbSpaces ausgegeben, die nicht zur Anlage der Tabellen benötigt werden. Alle Größenangaben liegen in der Einheit Kilobyte vor.

Die Größenangaben beschreiben den Platz in KB, der für die einzelnen Tabellen−Fragmente benötigt wird (***TableSpaces***), den Platz, der für die Systemkatalogtabellen benötigt wird (***Dictionary***) und den Platz, der für die interne Verwaltung, d.h. für die Partition−Pages (***Overhead***) benötigt wird. In der Spalte ***Total*** wird die Summe der drei Größenangaben abgebildet. Zur Kontrolle zeigt **ifddt** in der Spalte ***Free*** den freien Platz innerhalb des jeweiligen DbSpaces an. Ist der benötigte Platz zur Anlage der Tabellen nicht mehr frei, so sollten Sie von der Durchführung des Schemas Abstand nehmen und erst eine Umverteilung der Tabellen, z.B. mit Hilfe der **Storage−Datei**, vornehmen.

Beachten Sie bei der Größenangabe, dass es sich bei dem freien Platz um den Platz auf dem Ausgangs−Server handelt. Sofern Sie die Daten auf einen anderen Server übertragen, müssen Sie manuell prüfen, ob dort der Platz zur Anlage der Tabellen noch frei ist. Bedenken Sie dabei auch, dass der benötigte Platz größer sein kann, als hier angezeigt, wenn das Ziel−System größere Page−Größen aufweist.

4.4 Standard−Constraint−Namen unterdrücken (−c)

Falls Sie die Standard−Constraint−Namen einer Datenbank übernehmen, so können später Probleme bei der Anlage von Constraints auftreten, da sich die Standard−Constraint−Namen immer aus einem Buchstaben, der eindeutigen Tabellennummer und der eindeutigen Constraint−Nummer zusammensetzen. Haben Sie in der Vergangenheit eine Tabelle gelöscht, so werden bei der erneuten Tabellenanlage alle nachfolgenden Tabellen eine neue, kleinere, eindeutige Tabellennummer aufweisen. Bei Übernahme der Constraint−Namen bleiben aber die Tabellennummern innerhalb der Constraint−Namen erhalten. Neue Tabellen können somit eine Tabellennummer erhalten, die dann in Konflikt mit einer ehemaligen Tabellennummer gerät. Zwar sind die Tabellennummern konfliktfrei, nicht aber die übernommenen Constraint−Namen. Durch Angabe der Option **−c** sorgen Sie dafür, dass die Standard−Constraint−Namen nicht übernommen werden. Durch diese Option wird die Ausgabe der Standard−Constraint−Namen unterdrückt.

4.5 Not−Null−Constraint−Namen unterdrücken (−C)

Neben der Möglichkeit, Standard−Constraint−Namen zu unterdrücken, können Sie auch generell auf die Ausgabe der Constraint−Namen bei Not−Null−Constraints verzichten. Diese Option ist sinnvoll, wenn die meisten Ihrer Tabellenfelder Not−Null−Constraints enthalten. Somit kann **ifddt** auf die Prüfung der Constraint−Namen gänzlich verzichten und die Erstellung des Schemas erfolgt meßbar schneller. Die Not−Null Definitionen werden natürlich dennoch ausgegeben; lediglich deren Namen werden fehlen.
Das Unterdrücken der Ausgabe der Not−Null Constraint−Namen aktivieren Sie mit Hilfe der Option −C. Bei großen Datenbanken sollte diese Option standardmäßig aktiviert werden.

4.6 Erzeugen einer Standard storage Datei (−S)

Zur Anlage einer **Storage−Datei** wählen Sie anstelle der Option **−s** die Option **−S** gefolgt von einem Dateinamen. **ifddt** legt dann die Storage−Datei an, sofern die Datei bislang nicht existierte. Eine Datei−Endung ist nicht bindend; zur besseren Unterscheidung sollten Sie aber eine einprägsame Endung wählen. Sie können eine einzige Storage−Datei für mehrere Datenbanken nutzen, da innerhalb der Storage−Datei der Datenbankname zur zusätzlichen Unterscheidung der Tabellen− und Indexeinträge verwendet wird. Sollte der Dateiname bereits existieren, so aktualisiert **ifddt** lediglich die Storage−Datei. Die alte Storage−Datei wird dann im gleichen Pfadnamen mit der Endung *.bak* gespeichert.
Da Sie oftmals bei der ersten Erstellung der Storage−Datei kein Schema benötigen, sollten

Sie die Ausgabe von **ifddt** in den Papierkorb umlenken.
Verwenden Sie bei der ersten Erstellung der Storage−Datei auf keinen Fall die Option **−N**, sofern Sie beabsichtigen, die Storage−Datei anschließend zu ändern. Es würden unnötigerweise alle Tabellen entladen werden!

```
$ ifddt −d stores7 −S munich_shm.ddt > /dev/null
```

Bei der Erstellung der Storage−Datei werden alle server−spezifischen Eigenschaften aller Tabellen beschrieben. Die Spalte, die das monatliche Wachstum der Tabellen enthält, wird bei der Generierung mit dem Wert 0 vorbelegt. Existierte die Storage−Datei bereits, und enthielt diese Datei nur Einträge anderer Datenbanken, so bleiben die Einträge der anderen Datenbanken ebenso wie eventuell vorgenommene Kommentare von der Erstellung unberührt.

4.7 Extent−Multiplikator festlegen (−e/−E factor)

Sofern Sie entweder die Option **−s** oder die Option **−S** Storage−Datei nutzen, können Sie die aktuellen Extent−Größen für alle Tabellen pauschal mit einem Extent−Multiplikator multiplizieren. Der Extent−Multiplikator kann eine Genauigkeit von 7 Stellen aufweisen. Die resultierende Extent−Größe wird dabei generell auf ein Vielfaches einer Page−Größe gerundet. Die Obergrenze bei der Extent−Berechnung für die Extent−Größen ist auf 2GB bzw. 32GB, die Untergrenze für Extent−Größen ist auf 16 Kilobyte limitiert. Den Faktor geben Sie auf der Kommandozeile hinter der Option **−e** ein. Verwenden Sie zur Darstellung der Fließkommazahl anstelle des Kommazeichens den Punkt:

```
$ ifddt −d storesdemo −e 2.0 −s
```

Sofern Sie die Option **−S** Storage−Datei anwenden, werden auch alle betroffenen Tabellen in der Storage−Datei mit der neuen Extent−Größe aktualisiert. Ein wiederholter Aufruf mit der Option **−e** und dem gleichen Faktor würde daher die Extent−Größen erneut mit dem Faktor multiplizieren. Die Option **−e** bezieht sich immer auf die initiale Extentgröße. Die Standardgröße für die Next Extents entspricht einem Drittel der kalkulierten First Extent Größe und kann mit Hilfe der Option **−E** verändert werden. Auch hier gelten die Ober− und Untergrenzen für Extentwerte.

4.8 Zielversion festlegen (−u)

Mittels der Option **−u** können Sie die Versionsnummer der Informix Zielumgebung vorgeben. Diese Option ist sinnvoll, wenn Sie von der Version 7.x auf eine Version 9.x und

grösser migrieren und umgekehrt. Grundsätzlich prüft **ifddt** die Versionsnummer Ihrer aktuellen Umgebung. Wird dabei erkannt, dass die Zielversion nicht mit der aktuellen Version identisch ist, so werden alle zur Migration erforderlichen Schritte eingeleitet. Hierzu gehört die automatische Korrektur der Extent−Größen, wenn von der Version 7.x auf eine Version grösser gleich 9.x umgestellt wird. Beginnend mit der Version 9.x werden im Informix Dynamic Server alle Indizes als detached Indizes angelegt. Waren in Ihrer Version 7.x die Indizes noch attached, so müssen die Extent−Größen nach unten korrigiert werden. Zudem hat sich in der Version 9.x der Tabellenname *sysindexes* auf *sysindices* geändert. Dies wird bei der Korrektur der Extent−Größen der Systemtabellen berücksichtigt. Weitere Änderungen finden sich vor allem innerhalb der High Performance Loader Datenbank, bei den Datentypen, usw. Sofern Sie auf eine Version 11.50 migrieren, geben Sie in der Kommandozeile die Option wie folgt an:

```
$ ifddt −d storesdemo −s −u 11.50UC4
```

4.9 Entladen einer Datenbank (−N)

Beim Entladen legt **ifddt** ein Unterverzeichnis an, in dem die Dateien mit den Tabellendaten, sowie die Schema−Datei zur Wiederanlage der Datenbank gespeichert werden. Den Namen des Unterverzeichnisses wählen Sie selbst in der Kommandozeile hinter der Option −N. Sorgen Sie vor dem Aufruf dafür, dass das Unterverzeichnis noch nicht existiert und dass Sie die Benutzerrechte haben, um das Verzeichnis anzulegen. Die Tabellendaten werden in so genannte unload−Dateien entladen, deren Dateiname sich aus einem Teil des Tabellennamens und der eindeutigen Tabellennummer (tabid) zusammensetzt. Sofern Sie die Umgebungsvariable DELIMIDENT gesetzt haben, besteht der Tabellenname ausschliesslich aus der eindeutigen Tabellennummer. Die unload−Dateien weisen die Endung **.unl** auf. Die maximale Größe der unload−Dateien liegt bei 2130706432 Byte. Entstehen während der Entladevorgangs für eine Tabelle mehr Daten, so werden weitere unload−Dateien angelegt, deren Dateiendung dann um den Zusatz **.ex*n*** erweitert wird. Das Kürzel **.ex0** steht dabei für die erste Erweiterung, **.ex1** für die zweite Erweiterung, etc. Die maximale Größe der Dateien können Sie verkleinern, indem Sie vor dem Aufruf von **ifddt** die Umgebungsvariable MAX_FILE_SIZE auf die maximale Dateigröße in Byte setzen. Sie sollten die Größe aber nicht größer als 2GB definieren, falls es sich bei dem Zielsystem Ihres IBM Informix Servers um eine 32−bit Version handelt! Bei einem 64−bit Informix Server sollten Sie die Variable nicht größer setzen als 4GB. Haben Sie ein 64−bit ifddt Programm zusammen mit einem 64−bit Informix Server im Einsatz, so können Sie die Variable problemlos höher setzen. Möchten Sie eine Datenbank vollständig entladen, so müssen Sie zunächst ein Dateisystem

suchen, das noch hinreichend freien Platz zur Speicherung der entstehenden Dateien enthält. Stellt **ifddt** während des Entladens fest, dass nicht genügend freier Platz zur Verfügung steht, so bricht **ifddt** den Entladevorgang mit der Fehlermeldung:

```
tabelle.unl: No space left on device
```

beziehungsweise

```
I/O error while writing to file tabelle.unl
```

ab. In diesem Fall müssen Sie die zuletzt entladene Datei löschen und den Vorgang zusammen mit der Restart−Option des ifddt noch einmal starten. Stellen Sie zuvor sicher, dass sowohl die Umgebungsvariable MAX_FILE_SIZE richtig gesetzt ist und genügend Speicher in dem Dateisystem zur Verfügung steht.
Weist eine Tabelle einen CLUSTER INDEX auf, so werden die Tabellendaten sortiert entladen, so dass beim anschließenden Ladevorgang auf eine erneute Sortierung verzichtet werden kann. Während des Entladevorgangs prüft **ifddt** die entladenen Daten dahingehend, ob diese auch wieder geladen werden können.

```
Warning: double value possibly incorrect
Warning: conversion function problem near line n of 'xxx'
```

Tritt dieser Fehler auf, so bricht **ifddt** den Vorgang nicht ab. Prüfen Sie in diesem Fall manuell, ob die entsprechende unload−Datei in der betreffenden Zeile einen Fehler aufweist und versuchen Sie, die fehlerhaften Daten durch richtige zu ersetzen.

4.9.1 Restart Option (−w)

Die Restart−Option ermöglicht das Wiederaufsetzen nach einem Fehler beim Entladen von Datenbanken oder einer Reihe von Tabellen. Dabei prüft ifddt vor dem Entladen einer Tabelle, ob diese bereits im Entladeverzeichnis vorhanden ist. Indem Sie die zuletzt fehlerhafte Entladedatei entfernen, wird diese Tabelle erneut entladen. Die Option **−w** ist daher sowohl sinnvoll, wenn Daten einer Tabelle unvollständig entladen wurden, oder aber wenn das Dateisystem zu klein definiert war. Beachten Sie, dass bei dieser Option das entscheidende Kriterium für das erneute Entladen die Existenz der Entladedatei ist, und nicht die Anzahl der darin enthaltenen Datensätze!

4.9.2 Entladedateien komprimieren (−Z level)

Mittels der Option **−Z level** können Sie die Entladedateien direkt koprimieren. Je nach Kompression können die Daten beim Entladen auf 10% der Ausgangsgröße reduziert werden und lassen sich somit leichter in Dateisystemen speichern. Des weiteren erzielen Sie hierdurch den Vorteil, dass diese Entladedateien sich schneller über das Netzwerk übertragen lassen. Der *level* hinter der Option **−Z** legt das anzuwendende Kompressionsverfahren fest. Folgende Einstellungen sind standardmäßig vorgesehen:

1 Anwendung von gzip/gzip −d
2 Anwendung von compress/uncompress
3 Anwendung von zip/unzip
0 Anwendung der Programme, welche Sie in DDTCOMP definieren. DDTCOMP muss mit dem Kompressions− und Dekompressionsprogramm belegt werden, welche durch ein Semikolon voneinander getrennt sind, z.B.: **DDTCOMP="gzip;gzip −d"**

Während des Entladens werden die Kompressionsprogramme direkt seitens **ifddt** gestartet, d.h. die entstehenden Daten werden den Kompressionsprogrammen via Pipe übergeben. Die übrigen Regeln des Entladens, wie beispielsweise die High Performance Loader Optionen oder die ALTER TABLE Anweisungen bleiben hiervon unberührt. Beim Laden der Daten wird die Kompression berücksichtigt, d.h. beim Laden werden die Daten über die Dekompressionsprogramme ermittelt. Es wird davon ausgegangen, dass die Dekompressionsprogramme zum Zeitpunkt der Ausführung auf dem Zielsystem vorhanden sind. Ein Sicherstellung seitens **ifddt** erfolgt hierbei nicht. Stellen Sie daher bereits vor dem Entladen sicher, dass das Programm zur Dekompression auf dem Zielsystem vorhanden ist.

4.9.3 Umgebungsvariable *DBMONEY*

Bis zur Version 3.05 wurde die Umgebungsvariable DBMONEY seitens ifddt nicht ausgewertet, d.h. es wurde bei Fliesskommawerten wie SMALLFLOAT und FLOAT grundsätzlich ein Dezimalpunkt ausgegeben. Der Inhalt der Variable wird seit 3.05UD2 berücksichtigt und daher muss auf der Zielmaschine vor dem Ladevorgang sichergestellt werden, dass die Umgebungsvariable beim Laden den gleichen Wert aufweist, wie zum Zeitpunkt des Entladens.
Um während des Entladens garantieren zu können, dass nur logisch konsistente Daten gelesen werden, sperrt **ifddt** die zu entladende Datenbank während des ganzen Vorgangs exklusiv. Stellen Sie daher vor dem Aufruf von **ifddt** sicher, dass kein anderer Anwender zu diesem Zeitpunkt die Datenbank ausgewählt hat. Diese exklusive Sperre lässt sich mit Hilfe der Option **−l** umgehen. Jedoch ist die logische Konsistenz der Tabellendaten nach der

Übernahme nicht gewährleistet.
Beim Entladen schreibt **ifddt** die SQL−Anweisungen zur Anlage der Datenbank nicht auf die Standardausgabe, sondern erzeugt innerhalb des gewählten Unterverzeichnisses eine Anweisungsdatei mit dem Namen *datenbank.sql*. Diese Anweisungsdatei enthält abschließend nicht nur die SQL−Anweisungen zur Anlage des Datenbankschemas, sondern auch die entsprechenden Lade−Anweisungen zum Laden der einzelnen Tabellen. Wurden für eine Tabelle mehrere unload−Dateien erzeugt, so findet man in der Anweisungsdatei für diese großen Tabellen entsprechend mehrere Lade−Anweisungen.
Beachten Sie bitte, dass die Anweisungsdatei nicht den Befehl zur Anlage der Datenbank enthält. Wollen Sie mit Hilfe der Anweisungsdatei die Datenbank in einem anderen oder im gleichen Server erneut anlegen, so müssten Sie die Datenbank vorab erst manuell anlegen.
Beim Entladen können Sie − mit Ausnahme der Optionen **−t** und **−n** − alle weiteren Optionen anwenden. Ein typisches Beispiel könnte folgendermaßen aussehen:

```
$ rm −r /tmp/st.exp
$ ifddt −d stores7 −e 2 −S munich_shm.ddt −N /tmp/st.exp
```

Entladen wird dabei die Datenbank namens *stores7* in das Verzeichnis */tmp/st.exp*. Alle in der Storage−Datei ***munich_shm.ddt*** definierten Größen und DbSpace−Angaben werden im generierten SQL−Skript berücksichtigt und die Extent−Angaben werden mit dem Faktor 2.0 multipliziert. Neben den Befehlen zur Anlage und den Befehlen zum Laden der Tabellen befinden sich anschließend in der Datei ***/tmp/st.exp/stores7.sql*** auch alle weiteren Befehle zur Anlage der Views, Trigger, Rechte, Synonyme, Constraints, Stored Procedures und Indizes.
Um eventuellen Mißverständnissen vorzubeugen, sei an dieser Stelle darauf hingewiesen, dass der Aufbau und Inhalt des erzeugten Verzeichnisses inkompatibel zu dem vom Informix Kommando »dbexport« erzeugten Verzeichnis ist. Es ist somit nicht möglich, mittels »dbimport« die Datenbank wieder zu laden. Die generierte Anweisungsdatei enthält neben reinen SQL−Anweisungen auch Lade−Anweisungen, die ausschließlich vom Informix Programm »dbaccess« verstanden werden.

4.9.4 Alphanumerische Werte konvertieren (−k datei)

Mittels der Option **−k** können Sie für alle oder nur für bestimmte Spalten die darin enthaltenen Zeichen konvertieren. Dazu benötigen Sie eine Konvertierungsdatei, in der die Regeln für die Konvertierung festgelegt werden.
Eine Konvertierungsdatei besteht aus den umzusetzenden Zeichen, wobei stets in einer Zeile das umzusetzende Zeichen gefolgt von der Zeichenfolge Minus und Größer **(−>)** und dem Zielzeichen besteht. Wird vor den Umwandlungsregeln keine Tabellenspalte genannt, so gelten diese Umwandlungsregeln für alle alphanumerischen Spalten (ausgenommen sind

Spalten vom Datentyp TEXT).
Kommentare werden mit dem Raute−Zeichen '#' eingeleitet und gelten bis zum Zeilenende.
Soll eine Umwandlung nur für bestimmte Spalten gelten, so müssen Sie die Spaltennamen mit führendem Tabellennamen jeweils in eine gesonderte Zeile schreiben, die Sie mit einem Doppelpunkt abschliessen.
Die Umwandlung wirkt sich nur beim Entladen der Daten aus, d.h. auf diese Weise können Zeichensätze beim Entladen bereits angepasst werden. Soll eine Spalte zudem auf den Datentyp NCHAR bzw. NVARCHAR geändert werden, so fügen Sie hinter dem Doppelpunkt des Spaltennamens ein Plus−Zeichen (+) an. Der neue Datentyp wird in diesem Fall in der SQL−Datei angepasst.

```
#
# Konvertierungsdatei
#

#
# Allgemein soll gelten
#
        'a'     ->      'A'
        0x62    ->      0x42

#
# Nur für die nachfolgenden Tabellen gilt
#
adressen.ad_name:
adressen.ad_vorname:+
orte.or_name:

        'ä'     ->      0xFC
        'ö'     ->      0xFD
```

4.10 Verzögerte Indexanlage (−i und −O)

Ohne Angabe der Option **−i** werden die Indizes unmittelbar nach der Tabellendefinition ausgegeben. Die sofortige Anlage der Indizes behindert jedoch das Laden von Tabellen, weshalb Sie im Zusammenhang mit dem Kopieren, aber auch beim Entladen/Laden der Daten, die Indexanlage verzögern sollten.
Die verzögerte Indexanlage kann nun entweder gänzlich separat (**−i datei**), unmittelbar nach der jeweiligen tabellenspezifischen Datenübernahme (**−i** und **−O**) oder am Ende aller Datenübernahmen (**−i**) geschehen.

Werden die Indizes unmittelbar nach dem Laden der Daten angelegt, so verzögert sich verständlicherweise das Laden der nächsten Tabelle. Da in der Regel die Quell−Datenbank während der Datenübernahme gesperrt ist, bleibt die Quell−Datenbank nun bis zur Übernahme der letzten Tabelle gesperrt. Positiv wirkt sich jedoch auf die Indexanlage aus, dass die Tabellendaten sich nach dem Laden im Cache befinden.
Würden die Indizes separat oder am Ende des Anweisungsskripts erst generiert, so können zwar die Tabellendaten schneller kopiert werden, aber die Indexanlage im Zielsystem erfolgt erst anschliessend und dabei werden alle geladenen Tabellendaten erneut gelesen. Somit wird die gesamte Erstellung der Datenkopie länger dauern.
Wird neben der Option **−O** die Option **−i** in der Kommandozeile genannt, so werden nur die FOREIGN KEY Constraints am Ende des Skripts zusammen erstellt. Die übrigen Indizes würden frühzeitig, d.h. nach dem Laden der jeweiligen Tabelle generiert werden.
Damit auch FOREIGN KEY Constraints von dieser optimierten Indexanlage profitieren, empfehlen wir generell, keine impliziten FOREIGN KEY Indizes anzulegen. Besser ist die explizite Anlage eines Indexes auf den referenzierenden Spalten und die anschließende Anlage des Constraints.
*Seit der Version **ifddt 3.x** werden Cluster Indexes nicht mehr frühzeitig nach der Tabellenanlage erstellt, sondern ebenfalls erst nach dem Laden der Daten. Da jedoch die Daten bereits sortiert geladen werden, erfolgt nicht etwa der Befehl ALTER INDEX TO CLUSTER, sondern ein Update des Systemkatalogs, der ausschließlich vom Benutzer informix abgesetzt werden darf.*

4.11 Indizes reorganisieren (−I dateiname)

Mittels der Option **−I *dateiname*** können Sie die Indizes einzelner oder aller Tabellen reorganisieren. Da bei **detached** Indizes keine Extentgröße gewählt werden kann, und da die intern vergebene Größe in der Regel zu klein ausfällt, müssen Indizes von Zeit zu Zeit reorganisiert werden. Dabei ist es nicht notwendig, die ganze Tabelle zu reorganisieren; es reicht hier die einzelnen Indizes neu anzulegen.
Mittels der Option **−I *dateiname*** schreibt **ifddt** alle Anweisungen zum Löschen und zur Neuanlage aller Indizes in die Datei *dateiname*. Es ist dabei sichergestellt, dass zum einen alle referenzierenden Constraints, die implizit durch das Löschen von Primary Keys entfernt werden, auch wieder neu erstellt werden; zum anderen werden keine Constraintdefinitionen mehrfach gelöscht oder wieder angelegt. Die auf diese Weise erzeugte SQL Datei kann anschliessend manuell mittels *dbaccess* ausgeführt werden. Die gleichzeitige Anwendung der Option **−G** sorgt parallel zur Indexreorganisation dafür, dass implizite Indizes neu generiert werden.

4.12 Datenbanken kopieren (−n name)

Im Gegensatz zum Entladen einer Datenbank wird beim Kopieren einer Datenbank auf eine Zwischenablage der Daten verzichtet. Die Tabellendaten werden beim Kopieren direkt von der Ausgangs−Datenbank in eine Ziel−Datenbank übertragen. Für diese Übertragungsform müssen mehrere Voraussetzungen erfüllt sein.
Um eine solche Übertragung gewährleisten zu können, müssen sich die Ausgangs−Datenbank sowie die Ziel−Datenbank im gleichen Transaktions−Modus befinden. In der Regel werden beide Datenbanken ohne Transaktionssteuerung arbeiten bzw. wird der Modus vor dem eigentlichen Kopiervorgang mit Hilfe des Befehls:

```
$ ondblog nolog datenbankname
```

umgeschaltet. Sofern es sich bei der Zielversion um einen IBM Informix Dynamic Server der Version 7.31 bzw. 9.21 und größer handelt, besteht keine Notwendigkeit, den Transaktionsmodus eigens auf NO LOGGING umzuschalten; **ifddt** legt dabei alle Tabellen zunächst im raw−Modus an und ändert den Typ nach dem Laden der Tabellen automatisch.
Beide Datenbank−Systeme müssen sich entweder innerhalb einer IBM Informix Dynamic Server Instanz befinden, oder via einem Netz miteinander verbunden sein und eine direkte Kommunikation beider Server−Instanzen muss ohne Passwortabfrage möglich sein. Lesen Sie mehr hierzu im Abschnitt *»Administrative Anforderungen«*.

Zusammen mit der Option **−n** können Sie den Befehl zur Anlage einer neuen Datenbank erzwingen. Dazu müssen Sie einen der nachfolgenden Buchstaben unmittelbar hinter **−n** angeben.

Option	***resultierender Transaktionsmodus***
n	Ohne Logging
b	Buffered Logging
u	Unbuffered Logging
a	Unbuffered Logging Mode Ansi

Eine Option zur Bestimmung des Ziel−DbSpaces für die Datenbank existiert nicht. In der Regel ist es empfehlenswert, die Datenbank manuell mittels dem Werkzeug *dbaccess* anzulegen.

4.12.1 Clustered Indizes

Eine Eigenschaft von **ifddt** ist es, beim Erstellen des Anweisungs−Skripts zu prüfen, ob eine Tabelle einen CLUSTER INDEX aufweist. Die Anlage dieser Indizes sorgt stets dafür, dass die zugehörige Tabelle vollständig physikalisch reorganisiert wird. Diese Reorganisation dauert in der Regel länger, als das vorherige Laden der Tabelle und Anlegen der Indizes.
Um die physikalische Sortierung der Tabellendaten zu umgehen, werden die Daten bereits sortiert geladen. Allerdings ist es grundsätzlich bei einer INSERT−Anweisung, die einen SELECT enthält nicht möglich, der SELECT−Anweisung eine ORDER BY Klausel anzuhängen. Daher wählt **ifddt** eine etwas aufwendigere Variante; vor dem eigentlichen INSERT−Befehl wird in diesem Fall eine Stored Procedure namens **l_ifddt_sp()** definiert, welche die zu ladenden Daten sortiert zurück liefert. Unmittelbar nach dem Laden wird die Stored Procedure wieder gelöscht. Bei der Übernahme ganzer Datenbanken kann der Name der Stored Procedure nicht in Konflikt mit Ihren privaten Stored Procedures geraten, da Ihre Stored Procedures erst nach dem Laden der Tabellen definiert werden. Problematischer ist dies beim Kopieren einzelner Tabellen.
Sofern Sie die Parallel−Optionen (**−g**) anwenden, so werden die generierten Stored Procedure Namen um die Endung der jeweiligen Prozessnummer erweitert, um Konflikte bei den Namen der Stored Procedures zu vermeiden.

4.13 Tabellen reorganisieren (−n)

Sofern Sie nach der Option **−n** keinen Datenbanknamen angegeben und zudem die Option **−t** einsetzen, so geht **ifddt** davon aus, dass einzelne Tabellen innerhalb einer Datenbank reorganisiert werden sollen. In der Folge werden die SQL Befehle analog zur Datenbankkopie generiert, zusätzlich wird dabei aber berücksichtigt, dass die *alten Tabellen* abschliessend gelöscht werden. Es erfolgt somit eine Reorganisation in neue Tabellen mit finalem Löschen der Quelltabellen. Die Anwendung der Option **−n** ohne Angabe der Option **−t** ist unzulässig.

4.14 Umgebungsvariable INFORMIXSERVER setzen

Vor dem Erstellen des SQL−Skripts sollten Sie bedenken, dass die Daten später seitens des Ziel−Systems vom Ausgangs−System angefordert werden. Das heißt, dass Sie auf jeden Fall vor dem Aufruf von **ifddt** die Umgebungsvariable INFORMIXSERVER auf den Server−Namen des Servers setzen müssen, der die Ausgangs−Datenbank enthält.
Im ersten nachfolgenden Beispiel gehen wir davon aus, dass die Datenbank namens *stores7* vom Datenbank−Server ***berlin_tli*** des Rechners ***berlin*** auf den Datenbank−Server ***muc_soc***

des Rechners ***munich*** kopiert werden soll. Zur Erstellung des Anweisungsskripts müssen Sie dabei folgende Kommandos auf dem Rechner ***munich*** absetzen:

```
$ INFORMIXSERVER=berlin_tli
$ ifddt −d stores7 −s −i −c −C −n new > stores7.sql
```

Ist das Anweisungsskript einmal erstellt, so können Sie alle weiteren Schritte auf dem Ziel−System durchführen. Hierzu gehört nun noch das Anlegen der Datenbank und die Durchführung des Anweisungsskripts. Gehen wir davon aus, dass die Ausgangs−Datenbank bereits ohne Transaktionssteuerung arbeitet, so müssen also zum Kopieren der Datenbank noch folgende Schritte durchgeführt werden.

```
$INFORMIXSERVER=muc_soc
$echo 'create database stores7 in dbspace1' | dbaccess −
$dbaccess −e stores7 stores7 > stores.out 2> stores.err
```

Da wir beim Erzeugen des Anweisungsskripts die Option **−s** angegeben haben, wird das Anweisungsskript die DbSpace−Namen des Ausgangs−Systems übernehmen. Damit die Übernahme erfolgreich abläuft, ist es daher erforderlich, dass im Ziel−System die gleichen DbSpaces vorhanden sind und hinreichend freien Speicherplatz für die Tabellen aufweisen.
Dennoch ist es sinnvoll, die beim Kopieren der Datenbank möglicherweise entstehenden Fehlermeldungen in einer Datei zu protokollieren und abschließend zu analysieren. Suchen Sie in den Protokolldateien **stores.out** und **stores.err** nach allen Stellen, die das Wort ***Error*** enthalten.
Läuft alles fehlerfrei ab, so existiert abschließend eine Datenbank namens *stores7* im Datenbankserver *muc_soc*, die alle Daten der Datenbank *stores7* des Datenbankservers ***berlin_tli*** enthält.

4.15 Kopieren einer Datenbank innerhalb eines Servers

Möchten Sie eine Datenbank innerhalb einer Server Instanz kopieren, so bedenken Sie vorab, dass die neue Datenbank in anderen DbSpaces liegen sollte. Es ist in der Regel schneller, eine Datenbank von einer auf eine andere Festplatte zu kopieren, als innerhalb einer Festplatte. Weisen Sie **ifddt** daher an, für die neue Datenbank andere DbSpaces einzusetzen, sofern in Ihrem System ein DbSpace einer Festplatte entspricht. Soll die Anzahl der DbSpaces identisch bleiben, so empfehlen wir die Option **−m** zum DbSpace−Mapping; anderenfalls empfehlen wir den Einsatz der Option **−S** Storage−Datei.
Soll die Datenbank *stores7* auf dem Datenbankserver *muc_shm* unter dem Namen *st7c*

kopiert werden, so setzen Sie folgende Befehle ab:

```
$ export INFORMIXSERVER=muc_shm
$ ifddt −d stores7 −s −m −c −C −i −n st7c > st7c.sql
$ echo 'create database st7c in dictdbs' | dbaccess −
$ dbaccess −e st7c st7c > st.out 2> st.err
oder
$ export INFORMIXSERVER=muc_shm
$ echo 'create database st7c in dictdbs;' | dbaccess −
$ ifddt −d stores7 −s −m −c −C −i −n st7c ) | dbaccess st7c
```

Prüfen Sie auch nach diesen Befehlen wieder die Protokolldateien hinsichtlich eventuell aufgetretener Fehler. Prüfen Sie, ob in den Protokolldateien das Wort Error auftritt und prüfen Sie diese Stellen.

4.16 Tabellen selektieren (−|+t)

Die Option **−t tabellenmuster** dient dazu, nur bestimmte Tabellen aus dem Datenbankschema zu übernehmen. Alternativ können Sie durch die Option **+t tabellenmuster** auch die Negation der Option **−t tabellenmuster** erreichen. Das Tabellenmuster erlaubt alle Muster, welche Sie auch beim SQL−Vergleich mittels MATCHES verwenden können. Sofern Sie beide Varianten kombinieren, werden alle unter **−t** genannten Tabellen kopiert mit Ausnahme derer, welche dem Muster von **+t** entsprechen. Die gewünschten Tabellen können Sie in der Kommandozeile unmittelbar hinter der Option **−t** angeben. Möchten Sie mehrere Tabellen entladen, so können Sie die Option **−t tabelle** auch mehrfach nennen. Die Option **−t** bzw. **+t** zum Entladen einzelner Tabellen können Sie einerseits dazu benutzen, um bestimmte Tabellen in eine Datenbank neu aufzunehmen, oder aber um bestehende Tabellen einer Datenbank durch eine neuere Version zu ersetzen. Beim Ersetzen wird es erforderlich, die bestehende Tabelle zuvor aus der Datenbank zu entfernen; werden die Tabellen nur neu hinzugefügt, so sind keine weiteren Schritte als die Tabellenanlage erforderlich. *Siehe mehr hierzu unter Option **−r bzw. −R**.*

4.16.1 Tabellenrechte nicht unterdrücken (−p all)

Beachten Sie beim Entladen einzelner Tabellen, dass im Gegensatz zum Entladen ganzer Datenbanken, weder die tabellenspezifischen Benutzerrechte noch die Role−Definitionen ausgegeben werden. Sie müssen **ifddt** mit Hilfe der abschließenden Option **−p all** dazu anweisen, die Rechte auszugeben. Mit dieser Option werden jedoch auch die Role−Definitionen ausgegeben, was auf der späteren Ziel−Datenbank zu einem Fehler führen würde, sofern dort die Role−Definitionen bereits existieren. Bitte beachten Sie, dass die

Option **−p all** innerhalb der Kommandozeile stellungsorientiert ist. Nennen Sie diese Option stets am Ende der Kommandozeile.

4.16.2 Metazeichen in Tabellennamen

Möchten Sie mehrere Tabellen entladen, deren Namen gleiche Bestandteile aufweisen, so ist es auch möglich, die gleichen Teile zu nennen und für die ungleichen Teile Metazeichen anzugeben. Ein Stern (*) steht dabei für beliebig viele beliebige Zeichen, ein Fragezeichen (?) steht für genau ein beliebiges Zeichen. Metazeichen können Sie an jeder beliebigen Stelle des Tabellennamens und in beliebiger Kombination einsetzen. Sorgen Sie dafür, dass die genannten Metazeichen nicht von Ihrer Shell interpretiert werden. Setzen Sie daher den Tabellennamen innerhalb von Anführungszeichen, in Apostroph−Zeichen oder entwerten Sie die Bedeutung des Meta−Zeichens, indem Sie dem Metazeichen einen Schrägstrich (\ ) voranstellen.

```
$ ifddt −d sysmaster −t 'arc_*' −N arc.exp
```

Dieser Befehl weist **ifddt** an, nur die Tabellen der sysmaster Datenbank in das Verzeichnis arc.exp zu entladen, deren Tabellenname mit den Buchstaben arc_ beginnt, und beliebig viele weitere Zeichen enthält. Grundsätzlich ist es ein Unterschied, ob Sie die Option **−t** nennen oder nicht. Wenn Sie die Option **−t** '*' nennen, so werden im Gegensatz zum Aufruf ohne die Option **−t** keine Stored Procedure Definitionen, keine Rechte oder View−Definitionen ausgegeben.

4.16.3 Zugehörige Views entladen (−v)

Standardmäßig erzeugt **ifddt** sowohl beim Entladen als auch beim Kopieren einzelner Tabellen keine View−Definitionen. Sollen die View−Definitionen, die in direktem Zusammenhang mit den zu entladenden Tabellen stehen, übernommen werden, so geben Sie auf der Kommandozeile die Option **−v** an. Diese Option wird automatisch aktiviert, wenn Sie in der Kommandozeile entweder die Option **−r** oder **−R** angegeben haben, da durch das vorherige Löschen der Tabellen auch implizit alle damit in Verbindung stehenden View−Definitionen gelöscht werden.

4.16.4 Tabelle vor der Anlage löschen (−r/−R)

Bei der Übernahme einzelner oder aller Tabellen in eine bestehende Datenbank, würden die Befehle zur Anlage der Tabellen fehlschlagen, sofern die Tabellen auf dem Zielsystem bereits existieren. Sofern Sie sicher sind, dass die Tabellen auf dem Zielsystem bereits unter

gleichem Namen existieren, können Sie die alten Tabellen des Zielsystems vor der erneuten Anlage löschen. Mit Hilfe der Option **−r** weisen Sie **ifddt** dazu an, den entsprechenden DROP TABLE Befehl vor dem CREATE TABLE Befehl auszugeben. Sollten Sie jedoch nicht sicher sein, ob die Tabellen tatsächlich existieren, so verwenden Sie anstelle der Option **−r** die Option −R. Diese Option ist zwar langsamer als die Option −r, sie prüft jedoch vor dem DROP TABLE Befehl, ob die entsprechende Tabelle vorhanden ist und vermeidet auf diese Weise Fehlermeldungen in der Ausgabe.

4.16.5 Stored Procedures ausgeben (−f)

Werden einzelne Tabellen kopiert oder entladen, so fehlen im Ausgabeskript alle Stored Procedures, da es keine direkte Abhängigkeit zwischen Stored Procedures und den darin angesprochenen Tabellen gibt. Sollen diese ebenfalls mit ausgegeben werden, müssen Sie in der Kommandozeile die Option **−f** angeben. Diese Option bewirkt zugleich, dass die Stored Procedure spezifischen Rechte mit ausgegeben werden. Hierzu müssten Sie ebenfalls wieder die Option **−p all** am Ende der Kommandozeile angeben.

4.17 Tabellen kopieren

Das Kopieren einer oder mehrerer Tabellen unterscheidet sich vom Entladen der Tabellen dadurch, dass die Tabellendaten nicht in einzelne unload−Dateien entladen, sondern unmittelbar nach der Anlage direkt in die neuen Tabellen übernommen werden. Beim Kopieren einzelner Tabellen müssen die Transaktionsmodi beider beteiligten Datenbanken − analog zum Kopieren von Datenbanken − übereinstimmen. Im übrigen können Sie alle im Abschnitt »Datenbanken kopieren« genannten Optionen auch beim Kopieren von Tabellen verwenden.

4.18 Transformationen (−F datei)

Beim Entladen von Tabellen oder ganzer Datenbanken können Sie anstelle des Entladens der jeweiligen Tabelle auch freie SELECT Anweisungen festlegen. Dies ist beispielsweise beim Neuaufbau einer Datenbank sinnvoll, wenn Stammdaten aus anderen Tabellen geladen werden sollen, oder wenn beim Entladen nur bestimmte Spalten benötigt werden. Hierzu legen Sie eine Steuerdatei an, in welcher für jede zu transformierende Tabelle eine eigene SELECT Anweisung, oder eine Liste der zu entladenden Spalten definiert wird. Die SELECT Anweisung darf beliebig geartet sein, muss jedoch zumindest prinzipiell Sätze zurückliefern können. Die nachfolgende Tabelle beschreibt einen möglichen Aufbau.

```
#
# Transformation file Statistic Export
#
buchung:s:buchung.unl:select * from buchung where mod(buch_id,10)=7
deltasec:c:deltasec.unl:seq,reason:delta > 1
...
```

Kommentarzeilen beginnen mit einer Raute. In der ersten durch Semikolon getrennten Spalten befindet sich der Name der zu entladenden Tabelle, gefolgt von einem **s**, wenn es sich nachfolgend um eine SELECT Anweisung handelt. Oder um ein **c**, wenn nachfolgend eine Auflistung der zu selektierenden Spalten folgt. Die dritte Spalte enthält den Pfadnamen der Datei, in welche die Daten entladen werden. Im Falle einer Spaltenauflistung enthält die letzte Spalte eine optionale WHERE Klausel. Beim Entladen werden die hier genannten Tabellen nicht standardmäßig entladen, sondern entsprechend den hier definierten Vorgaben. Alle übrigen Tabellen würden regulär entladen.

```
$ ifddt −d st7c −F ste.etl −t buchung −t deltasec −c −C −s −N ste.exp
```

4.19 Allgemeine Optionen

Neben den bereits beschriebenen Optionen existieren weitere nützliche Optionen, die Sie bei Bedarf einsetzen können:

4.20 Benutzernamen unterdrücken (−a)

Befindet man sich nicht gerade in einer ANSI Datenbank, so lässt man in der Regel bei der Anlage einer Tabelle den Namen des Tabelleneigentümers weg, da man selbst der Eigentümer ist und natürlich auch seitens der Datenbank wird. Bei der Generierung des Datenbankschemas jedoch wird der Eigentümer einer jeden Tabelle, eines jeden Indexes sowie einer jeden Stored Procedure stets mit ausgegeben.
Sofern Sie beabsichtigen, das Schema auf eine andere Maschine zu übertragen, auf der die Benutzer des lokalen Systems nicht vorhanden sind, so könnte die Übernahme der Rechte ernsthafte Probleme auf der Zielmaschine verursachen.
Um dieses Problem zu vermeiden, können Sie die Ausgabe der Benutzernamen bei der Anlage von Objekten unterdrücken, indem Sie auf der Kommandozeile die Option **−a** absetzen.
ifddt wird bei der Angabe dieser Option grundsätzlich alle Benutzernamen unterdrücken. Somit entfällt auch die Angabe der entsprechenden Benutzerrechte. Lediglich der Benutzername »informix« und »public« werden übernommen und somit auch deren

spezifische Rechtezuweisungen.
Benutzerangaben innerhalb von Views werden nur analysiert und entfernt, sofern Sie neben der Option **−a** die Umgebungsvariable **DDTINVIEW** auf einen Wert größer oder gleich 1 gesetzt haben.
Diese Option steht nicht im Widerspruch zur nachfolgend erklärten Option **−o owner**.

4.21 Bestimmte Benutzertabellen ausgeben (−o owner)

Möchten Sie aus einer Datenbank alle Tabellen in eine andere Datenbank übernehmen, die einem bestimmten Benutzer gehören, so können Sie die Option **−o owner** einsetzen, wobei Sie *owner* durch den Benutzernamen ersetzen. Diese Option wirkt nicht selektierend wie die Option **−t**; wird die Option **−t** nicht mit angegeben, so werden alle übrigen Objekte der Datenbank (Stored Procedures, etc.) mit übernommen. Wollten Sie dies vermeiden, so wählen Sie die zusätzliche Angabe der Option **−t '*'**.

4.22 Views, Trigger, Procedures und Synonyme auslagern (−b datei)

In Umgebungen mit mehreren Datenbankservern ist es üblich, Views und Synonyme zu definieren, die sich auf externe Datenbanken beziehen. Ebenos können Stored Procedures Anwendung finden, die in anderen Datenbanken definiert wurden.
Eine Übernahme einer solchen Datenbank in eine andere wird Probleme aufwerfen, wenn diese vom Ziel−System aus nicht erreichbar ist, oder wenn im Zielsystem die externe Datenbank anders benannt wird oder erst später übernommen wird. Bei der Übernahme würden zwangsläufig die Befehle zur Anlage der Trigger, Views und Synonyme fehlschlagen.
Um diesen Problemen vorzubeugen erlaubt **ifddt** mit Hilfe der Option **−b *dateiname*** die Anlage aller Stored Procedures, Views, Trigger und Synonyme auszulagern, so dass diese erst später in einem separaten »dbaccess«−Lauf erzeugt werden. Die Auslagerung erfolgt in eine eigenständige Datei, die Sie anschliessend mit einem gewöhnlichen Editor manipulieren können.
Diese Option kann nicht beim Entladen mittels der Option −N angewendet werden !

4.23 Schnelle Indexanlage (−i)

Grundsätzlich ist es bei Tabellen empfehlenswert, die Indizes erst nach dem Laden der Daten

anzulegen. Dies trifft sowohl für explizite Indizes zu, die mittels der CREATE INDEX Anweisung erzeugt werden, als auch für implizite Indizes, die mit Unique−, Primary− und Foreign Key Constraint Definitionen einher gehen.
Durch Angabe der Option **−i** veranlassen Sie **ifddt** dazu, die Befehle zur Indexanlage an das Ende der Ausgabedatei zu legen. Dabei werden auch alle impliziten Indizes, die durch Constraint Definitionen entstehen, an das Ende verlagert. Die Stelle innerhalb der Ausgabedatei können Sie leicht auffinden; sie wird mit dem Kommentar:

```
{ Index Creation }
```

gekennzeichnet. Unmittelbar danach erfolgt die SQL−Anweisung:

```
set pdqpriority 95;
```

Diese sorgt dafür, dass auf Maschinen mit hinreichend großem Speicher die PDQ−Optionen des Datenbankservers genutzt werden können. Es ist daher empfehlenswert, auf dem Ziel−Server den für PDQ− und Sortieraufgaben reservierten Speicher vor dem Laden entsprechend hoch zu setzen. Sie erreichen dies, indem Sie auf dem Ziel−Server vor dem Laden das Kommando onmode absetzen. Sofern das Erstellen der Datenbank auf dem Ziel−Server ein exklusiver Vorgang ist, können Sie bis zu 80% des bereit gestellten virtuellen Shared Memories für PDQ−Aufgaben einsetzen. Setzen Sie in diesem Fall folgende Kommandos ab:

```
$ onmode −M SpeicherInKilobyte
$ onmode −Q 1
```

SpeicherInKilobyte sollte dabei circa 0.8 x SHMVIRTSIZE sein.

4.24 Indizes auslagern (−i datei)

Sie können die Anlage expliziter Indizes grundsätzlich in eine separate Datei auslagern. Diese Option sollten Sie jedoch vermeiden, sofern Ihre Datenbank implizite Indizes aufweist. Die Option ist lediglich sinnvoll, wenn Sie die expliziten Indizes in irgendeiner Form manuell nachbearbeiten, z.B. fragmentieren möchten. Vermeiden Sie das Auslagern der Indizes auch dann, wenn Sie die Datenbank oder Tabellen entladen, da die Zieldatei für die ausgelagerten Indizes sich nicht in dem für das Entladen erzeugten Unterverzeichnisses befinden kann. Zum Auslagern der Indizes müssen Sie im Anschluss an die Option **−i** den Pfadnamen für die Indexbefehle angeben.

4.25 Indizes reorganisieren (−I datei)

Bisweilen werden Sie in Ihren Informix Instanzen feststellen, dass − obgleich Sie korrekte Extentgrößen für Ihre Tabellen definiert hatten, die Anzahl der Index−Extents schnell ansteigt. Ursache hierfür kann eine fragmentierte Tabelle oder aber viele nicht gefüllte *variabel lange Objekte* sein. Möchten Sie bestehende Indizes einer Datenbank oder verschiedener Tabellen reorganisieren, so setzen Sie die Option **−I dateiname** ein. Alle zur Indexreorganisation notwendigen Befehle werden dabei in der genannten Datei abgelegt, wobei folgendes gilt:
Wurde ein Index in Folge eines Referentiellen Constraints angelegt und nicht speziell benannt, so wird zunächst der Constraint gelöscht und anschliessend wieder angelegt. Durch die Wiederanlage des Constraints wird auch der Index neu erstellt. Handelt es sich um einen Primär− oder Unique Constraint, so werden durch das Löschen dieses Constraints implizit alle referenzierenden Constraints gelöscht. Aus diesem Grund werden alle davon betroffenen referenzierenden Constraints am Ende der Datei ebenfalls neu erstellt.
Handelt es sich um einen einfachen Index, so wird dieser zunächst gelöscht, und anschliessend erneut angelegt. Um sicherzustellen, dass während der Verarbeitung kein Problem auftritt, werden alle diese zur Reorganisation erforderlichen Aktionen in einer einzigen Transaktionsklammer plaziert. Stellen Sie vor dem Aufruf dieser Befehlsfolge daher sicher, dass Sie vorab die Umgebungsvariable DBACCNOIGN auf den Wert 1 gesetzt haben. Somit wird die abschliessende Anweisung **commit work** im Fehlerfall nicht durchgeführt.

Grundsätzlich werden zunächst die Constraints, dann die explizit benannten Indizes gelöscht. Anschliessend erfolgt die Wiederanlage der benannten eindeutigen Schlüssel, gefolgt von der Anlage der Primary− und Unique−Constraints. Abschliessend erfolgt dann die Anlage der referenzierenden Constraints.

4.26 Constraints auslagern (−B datei)

Nach dem Kopieren bzw. Laden von Tabellen wird viel Zeit für die Wiederanlage von Fremdschlüsselbeziehungen verwendet, obgleich eine Prüfung eben dieser in der Regel entfallen kann. Falls Sie sicher sind, dass eine Wiederanlage dieser *Constraints* sichergestellt ist, können Sie die Constraintdefinitionen in einer separaten Datei auslagern. Setzen Sie hierzu die Option **−B** gefolgt von einem Dateinamen ein; beim Entladen *Option −N* sollten Sie ein vom Entladeverzeichnis abweichenden absoluten Pfadnamen verwenden.
Die dabei erzeugte Constraintdatei ist eine ASCII Datei, welche einzelne, durch ein Semikolon voneinander getrennte Spalten aufweist.

```
#########################
## KEY CONSTRAINT FILE ##
## Ensure PDQ = low    ##
#########################
stefan:prim:2:::::f1:
stefan:prim1:2:::::f1:
stefan:tsollkvpv:2:::stvorpah:u186_173:tsollkvpv_id:
stefan:b1666_prim:2:::::f1:
stefan:b1666:4:::::f1:
stefan:b1666:0:stefan:b1666_prim:::f1:
```

In der ersten Spalte befindet sich der Eigentümername der Tabelle, gefolgt vom Tabellennamen. Die dritte Spalte enthält eine **0**, wenn es sich um einen Fremdschlüssel, eine **4**, wenn es sich um einen eindeutigen Schlüssel, und eine **2**, wenn es sich um einen Primärschlüssel handelt. Die vierte Spalte enthält den Eigentümernamen des Constraints, auf den referenziert wird und die fünfte Spalte den referenzierten Constraintnamen. In der sechsten Spalte steht der Eingetümername des aktuellen Constraints gefolgt vom Constraintnamen in der siebten Spalte. Diese beiden Ausgaben erscheinen jedoch nur dann, wenn es sich um einen expliziten Constraintnamen handelt, d.h. wenn der Administrator einen expliziten Constraintnamen definiert hatte. Hinter der siebten Spalte folgen die einzelnen Spaltennamen in der Reihenfolge der Indexanlage.
Bei der Wiederanlage werden dann alle Constraints mittels des impliziten ifddt Werkzeugs ***ifcons*** angelegt. Dabei werden alle in der Datei genannten Foreign Key Constraints, für deren Spalten bereits ein entsprechender Index existiert, ad hoc angelegt ohne Prüfung, ob die Fremdschlüsselbeziehungen erfüllt sind. Wir weisen vorsorglich darauf hin, dass auf diese Weise Inkonsistenen auftreten können, sofern der Kopiervorgang nicht erfolgreich abgeschlossen wurde.

4.27 Implizite Indizes umgehen(−G)

Haben Sie bislang lediglich Primary− und Foreign−Key Definitionen abgesetzt, nicht aber auf den Spalten vorab Indizes definiert, so hat der Datenbankserver implizite Indizes erzeugt. Implizite Indizes unterscheiden sich von anderen Indizes vor allem darin, dass man sie nicht mit einem Indexnamen ansprechen kann. Die Indexnamen beginnen mit einem Leerzeichen und eben dies sorgt dafür, dass ein solcher Index nicht als Clustered Index angelegt, oder in einen anderen DbSpace verschoben werden kann. Auch das Löschen eines solchen Indexes ist ausgeschlossen, denn er wird implizit mit dem Constraint gelöscht. Hatte man zuvor für einen Constraint einen Index, und hat man diesen Index gelöscht, so entsteht ebenfalls ein impliziter Index, der explizite wird sozusagen nur umbenannt.

Für die Reorganisation ist der größte Nachteil aber wohl der, dass die Foreign−Key Constraints auch nach dem Kopieren der Daten vollständig neu angelegt werden müssen. Vor allem wenn Sie die Constraints auslagern empfiehlt es sich, implizite Indizes gleichzeitig zu entfernen. Setzen Sie auf der Kommandozeile die Option **−G** ein, so werden alle impliziten Indexnamen in explizite Indexnamen überführt, indem das führende Leerzeichen der Indizes durch den Buchstaben ***X*** ersetzt wird. Besonders profitieren Sie dann, wenn Sie zeitleich die Option **−B datei** einsetzen.

4.28 Indizes prüfen(−A datei)

Bisweilen ist es sinnvoll, bestehende Datenbanken hinsichtlich des Optimiererverhaltens zu überprüfen. Dies gilt besonders für Indizes und der Frage, ob sich der Optimierer in den verschiedenen Einstellungen (OPTCOMPIND 0/1/2) für die Verwendung des Indexes entscheidet. Mittels der Option **−A** können Sie eine solche Prüfung bei bestimmten Tabellen oder der ganzen Datenbank vornehmen. Das Resultat der Prüfung wird in der Datei protokolliert, welche Sie hinter der Option −A nennen. In der Ausgabedatei wird in jeder Zeile das Optimiererverhalten für ein Index beschrieben.

```
"informix".fraggy:fraggy_idx01:0:Y::
"informix".fraggy:fraggy_idx01:2:Y::
"stefan".tsollkvpv:tsollkvpv_id:0:Y::
"stefan".tsollkvpv:tsollkvpv_id:2:N:real # of rows in table is 0
"stefan".tsollkvpv:tsollkvpv_idx_731:0:Y::
"stefan".tsollkvpv:tsollkvpv_idx_731:2:N:real # of rows in table is 0
"stefan".tsollkvpv:tsollkvpv_idx_732:0:Y::
"stefan".tsollkvpv:tsollkvpv_idx_732:2:N:real # of rows in table is 0
"stefan".tsollkvpv:tsollkvpv_idx_733:0:Y::
"stefan".tsollkvpv:tsollkvpv_idx_733:2:N:real # of rows in table is 0
"stefan".tsollkvpv:tsollkvpv_idx_pers2:0:Y::
...
```

Die Ausgabezeilen sind durch Doppelpunkt voneinander getrennt. In der ersten Spalte befindet sich der Tabellenname mit Eigentümer gefolgt vom Indexnamen. In der dritten Spalte befindet sich die jeweilige Optimierereinstellung OPTCOMPIND=0 bzw. OPTCOMPIND=2. In der vierten Spalte befindet sich die Information, ob der Index seitens des Optimierers verwendet wird (Y=Yes N=No) und in der letzten Spalte ggf. ein Kommentar, was die mögliche Ursache für das Verhalten ist.

4.29 Zusatzoptionen bei Storage Dateien

Sofern Sie die Option **−S Storage−Datei** gewählt haben, können Sie die Zusatzoptionen −D dbspacelist zur Bestimmung der Ziel−DbSpaces sowie die Option −M month zur Berechnung der Extent−Größen anwenden.

4.29.1 DbSpace Definitionen festlegen (−D dbspacelist)

Sofern Sie mit einer Storage−Datei arbeiten möchten, und bei einer oder mehreren Tabellen die letzte Spalte der Storage−Datei nicht angeben, so wird **ifddt** die Tabellen in einem beliebigen, existierenden DbSpace anlegen. Beim Verteilen der Tabellen versucht **ifddt**, die vorhandenen DbSpaces so zu belegen, dass diese nach dem Anlegen der Tabellen gleichmäßig gefüllt sind.
Wenn Sie die Tabellen jedoch anschließend in ein anderes Informix Dynamic Server System übertragen, in dem die DbSpaces des Ausgangs−Systems nicht existieren, oder wenn die DbSpaces im Zielsystem nicht den gleichen freien Platz aufweisen, so ist es ratsam, die Namen und Größen der DbSpaces des Zielsystems bereits beim Aufruf von **ifddt** bekannt zu machen.
Die verschiedenen Werte werden in der Kommandozeile durch einen Doppelpunkt voneinander getrennt. Bei den Angaben muss es sich stets um ein Tupel bestehend aus DbSpace−Name und dem freien Platz in Kilobyte handeln. Eingeleitet wird die DbSpace Definition mit der Option −D, z.B:
Aufbau der Datei ***munich.ddt***

```
$ ifddt −d st7c −S munich.ddt −D 'sp1:200000:sp2:300000'
```

In diesem Beispiel werden alle in der Storage−Datei *munich−ddt* genannten Tabellen, bei denen in der letzten Spalte die Angabe des DbSpaces bzw. eines Fragmentierungs−Ausdrucks fehlt, in den DbSpaces *sp1* und *sp2* angelegt. Als Berechnungsgrundlage dient dabei die angegebene Größe der DbSpaces. Dabei versucht **ifddt** bei der Tabellenanlage dafür zu sorgen, dass abschließend alle DbSpaces gleichmäßig viel freien Platz aufweisen. Durch eine geschickte Angabe der Größen können Sie selbst dafür sorgen, dass ein DbSpace stärker oder weniger stark belegt wird. Sollten namensgleiche DbSpaces im Ausgangs−System vorhanden sein, so wirkt die Option in der Kommandozeile stärker.

4.29.2 Extent Berechnung anhand monatlicher Wachstumswerte (−M month)

In der vorletzten Spalte der Storage−Datei können Sie die eintragen, wieviele Datensätze in der jeweiligen Tabelle je Monat hinzukommen. Anhand dieses Wertes und den in der Kommandozeile angegebenen Monaten errechnet **ifddt** die für diese Tabelle maßgeblichen Extent−Größen. Diese Extent−Werte werden sodann in der Storage−Datei gespeichert. Vorsicht ist daher bei mehrmaligem Aufruf vom **ifddt** geboten, da sich mit jedem Aufruf die Basisgröße der Extents vergrößert.

4.30 DbSpace Mapping (−m)

Mittels der Option **−m** weisen Sie **ifddt** dazu an, anstelle der realen DbSpace−Namen beliebige andere DbSpace−Namen zu verwenden. Hierzu müssen Sie in der Datenbank **sysmaster** eine Tabelle namens **dbsmap** anlegen, die aus den Spalten ***newdbs*** und ***olddbs*** besteht. Die Tabelle muss unter der Benutzerkennung *informix* erstellt werden.

```
$ dbaccess sysmaster <<eof
create table dbsmap(
        olddbs varchar(128),
        newdbs varchar(128)
);
eof
```

Sie können die Tabelle dbsmap auch mit weiteren Spalten ausstatten und die Reihenfolge der Spaltendefinition spielt keine Rolle. Haben Sie die Tabelle dbsmap einmal erstellt, so können Sie diese mit beliebigen DbSpace−Namen füllen. Befindet sich eine Tabelle in einem DbSpace, dessen Name mit einem Wert der Tabellenspalte dbsmap.olddbs übereinstimmt, so verwendet **ifddt** bei der Ausgabe den Wert, den Sie in der Spalte dbsmap.newdbs eingetragen haben. Es spielt dabei keine Rolle, ob der entsprechende DbSpace im aktuellen Server existiert oder nicht. Es ist auch durchaus möglich, dass mehrere alte DbSpace−Namen auf einen gemeinsamen neuen DbSpace−Namen verweisen.
Diese Option ist äußerst praktisch, wenn Sie öfters Tabellen in ein anderes System übertragen müssen, das andere DbSpace−Namen aufweist.

4.31 Tabellenliste generieren (−y datei)

Möchten Sie beim direkten Kopieren eine Liste mit den Tabellen erstellen, welche letztlich kopiert werden, so können Sie auf der Kommandozeile die Option **−y** gefolgt von einem

Pfadnamen angeben. Die dabei generierte Datei kann später auch als Eingabedatei für das Programm **ifddtrows** dienen. Mittels **ifddtrows** lässt sich die übergebene Tabellenliste erweitern um die reale Datensatzanzahl je Tabelle. Diese Informationen sind bisweilen sinnvoll um ein erfolgreiches Kopieren von Tabellen zu prüfen. Das Trennzeichen für diese Datei ist stets das Pipe Symbol, unabhängig von der Umgebungsvariable **DBDELIMITER**.

4.32 Tabellensuche API (−z)

Die Option **−z** aktiviert eine zusätzliche Prüfung, ob Tabellen in einen Kopiervorgang übernommen werden sollen. Zunächst werden alle lokalen Regeln wie die Option **−t** bzw. **+t**, **−o** etc. angewendet. Alle Tabellen, die diesen Kriterien genügen, werden bei Angabe der Option **−z** zusätzlich an die Funktion ***ifddt_lookup*** der Datenbank **sysmaster** übergeben. Liefert diese Funktion einen Wert ungleich **0**, so wird die Tabelle weggelassen. Liefert die Funktion den Wert **0**, so wird die Tabelle verwendet. Der Aufbau einer solchen Interface Funktion sieht wie folgt aus:

```
dbaccess sysmaster <<eof
create procedure ifddt_lookup(
        dbname_vc varchar(128),
        tabname_vc varchar(128),
        owner_vc char(32)
        ) returning int;
        define rows_l int8;

        select sum(iswrites+isrewrites+isdeletes)
                into
                        rows_l
                from
                        sysptprof
                where
                        tabname = tabname_vc and
                        dbsname = dbname_vc;

        if rows_l > 0 then
                return 1;
        end if;
        return 0;
end procedure;
eof
```

Die Interface Funktion muss auf dem Quellsystem in der **sysmaster** Datenbank bereitgestellt und für alle Benutzer ausführbar gemacht werden. Beim späteren Aufruf wird der erste Parameter mit der zu kopierenden Datenbank belegt, der zweite Parameter enthält den voll

qualifizierten Tabellennamen und der dritte Parameter wird mit dem Namen des Tabelleneigentümers belegt, unabhängig davon, ob es sich um eine ANSI Datenbank handelt.

4.33 Update Statistics (−U)

Sowohl beim Kopieren, als auch beim Entladen von Tabellen und ganzen Datenbanken ist es sinnvoll, abschließend die Statistiken des Optimierers für die Tabellen mittels des Befehls UPDATE STATISTICS zu aktualisieren. Dieser Befehl sorgt innerhalb des Servers jedoch dafür, dass die jeweilige Tabelle vollständig inklusive aller Indizes gelesen wird. Daher ist es sinnvoll, die Statistiken sofort zu aktualisieren, sobald alle Informationen, die die Statistiken betreffen, geladen sind.
Weist eine Tabelle keinen Index auf, so kann die Statistik bereits nach dem Laden der Daten aktualisiert werden. Enthält die Tabelle ausschließlich explizite Indizes, so kann die Statistik nach der Erstellung des letzten Indexes erfolgen. Weist eine Tabelle implizite Indizes auf, so kann die Aktualisierung der Statistik erst nach der letzten Constraint−Definition erfolgen.
Mit der Option **−U** weisen Sie **ifddt** dazu an, die entsprechenden Befehle zur Aktualisierung der Statistiken auszugeben. Sofern der Datenbankserver einen hinreichend großen Cache Buffer Pool aufweist, wird durch die Platzierung der Befehle auf ein erneutes Lesen der Tabellen verzichtet.
Beachten Sie bitte auch, dass **ifddt** evtl. explizite Update Statistics High/Medium Befehle ebenso ausgibt, wie die Low Befehle.

4.34 Cache Optimierung (−O)

Die Option **−O** veranlasst **ifddt** dazu, die Befehle zur Indexanlage unmittelbar hinter evtl. Ladebefehle zu setzen und die zugehörigen Update Statistics Anweisungen unmittelbar hinter die letzte Indexanlage zu platzieren. Auf diese Weise kann bei Tabellen, deren Datenpages und Indexpages vollständig in den Cache Buffer passen, auf ein erneutes Lesen der Tabellendaten verzichtet werden. Eine zeitliche Einsparung ist folglich dann zu erwarten, wenn es sich überwiegend um Tabellen mittlerer Größe handelt.

4.35 Datenbanksperre beim Entladen vermeiden (−l)

Soll während eines Entladevorgangs keine exklusive Sperre auf der Quelldatenbank gesetzt werden, so setzen Sie die Option **−l** ein. Beachten Sie dabei jedoch, dass bedingt durch die fehlende Sperre inkonsistente Daten entladen werden können. Zu Inkonsistenzen kommt es dann, wenn während des Entladevorgangs Datensätze in einer noch nicht vollständig entladenen Tabelle eingetragen, gelöscht oder modifiziert werden.

4.36 High Performance Loader einsetzen (−H datenbank)

Zum Laden ihrer unload−Dateien können Sie alternativ zur load−Anweisung des Programms *dbaccess* auch den Informix High Performance Loader einsetzen. Mittels des High Performance Loaders werden Tabellen mit vielen, komplexen Datensätzen deutlich schneller in die Zieltabelle übertragen. Beim Aufruf des **ifddt** müssen Sie dazu die Option **−H** gefolgt vom Namen der Ziel−Datenbank angeben. Die Option **−H** ist zur Zeit nur in Kombination mit der Option **−N** zulässig, d.h. diese Eigenschaft können Sie nur in Zusammenhang mit dem Entladen von Tabellen oder ganzer Datenbanken einsetzen.
Nach dem Entladevorgang befinden sich im Entladeverzeichnis neben den unload−Dateien der Endung **.unl** und der SQL−Datei die Dateien **onpload.sql** und einige Dateien mit dem Prefix **HPL*xxx*.lod**.
Bevor Sie im Zielsystem den Ladevorgang einleiten, müssen Sie bei Anwendung dieser Option zunächst die High Performance Loader Datenbank anlegen. Starten Sie im Zielsystem daher zunächst folgenden Befehl:

```
$ cd entlade−verzeichnis
$ ./onpload.sh
```

Dieser Befehl erzeugt die Datenbank namens **ifddtpload**. Stellen Sie daher zuvor sicher, dass Sie keine gleichnamige Datenbank im Zielsystem haben. Setzen Sie nun die HPL Umgebungsvariable DBONPLOAD auf den Namen **ifddtpload**

```
$ DBONPLOAD=ifddtpload
$ export DBONPLOAD
```

Nun können Sie den Ladevorgang wie gewohnt starten. Nach dem Laden der Daten befinden sich im Verzeichnis /tmp für jede mit **onpload** geladene Tabelle zwei Dateien mit der Endung **.rej** und **.log**. Prüfen Sie die **.log** Datei hinsichtlich evtl. aufgetretener Fehler.
Die Anwendung des HPL ist nicht grundsätzlich schneller als der herkömmliche load−Befehl des Programms **dbaccess**. Lediglich bei großen Tabellen kann man von der Anwendung des HPL profitieren. Aus diesem Grund wird **ifddt** nur dann den Weg über den HPL wählen, falls die zu ladende Tabelle mehr als 10000 Sätze aufweist. Sofern Ihnen dieser Wert zu niedrig oder zu hoch erscheint, können Sie diese minimale Anzahl von Datensätzen mit Hilfe der Umgebungsvariablen MINHPLROWS vor dem Start von **ifddt** verändern.

```
$ MINHPLROWS=200000
$ export MINHPLROWS
```

In diesem Beispiel werden nur Tabellen, die mehr als 200000 Sätze enthalten, mittels dem HPL geladen, die übrigen werden regulär mit dem load−Befehl des **dbaccess** geladen.

Beachten Sie, dass weitere Kriterien gibt, welche einen schnellen Ladevorgang mittels des High Performance Loaders ausschließen. Hierzu zählt unter anderem eine Satzlänge die nicht mehr in eine Systempage passt, sowie bestimmte Datentypen, welche seitens des HPL nicht unterstützt werden.

Nach dem Ladevorgang mittels des High Performance Loaders muss eine Datensicherung mittels des Informix Dynamic Servers folgen, um ein fehlerfreies Arbeiten im Anschluss sicher zu stellen. Seitens der **ifddt** Werkezeuge wird lediglich eine Pseudo−Sicherung implementiert, um ein Weiterarbeiten möglich zu machen. Stellen Sie bitte bei Bedarf selbst noch einmal sicher, dass Sie eine vollständige Datensicherung nach dem Ladevorgang haben, um im Ernstfall ein Wiederaufsetzen möglich zu machen.

4.37 Optionen zur Parallelverarbeitung (−g)

Ladevorgänge lassen sich prinzipiell beschleunigen, indem mehrere Ladeprozesse gleichzeitig gestartet werden. Dies gilt nicht nur für Mehrprozessor−Systeme, sondern durchaus auch für Ein−Prozessor−Systeme. Beobachtet man während des Ladevorgangs die Systemauslastung, so lässt sich eine durchschnittliche Auslastung der Systemressourcen von etwa 50% ausmachen. Ursache hierfür sind die intern synchron ablaufenden Vorgänge, d.h es werden entweder Daten gelesen, oder verarbeitet, oder geschrieben. Beim Lesen und Schreiben von Daten, ist der Prozessor unbelastet, beim Verarbeiten der Daten sind die Festplatten unbelastet.
Um die Ressourcen besser zu nutzen, empfiehlt es sich, die Daten parallel zu laden und somit die Ressourcen besser zu nutzen. Dies gilt aber nur dann, wenn während des Ladevorgangs keine weiteren Benutzer das System belasten.
Mittels der Sub−Optionen von **ifddt** können Sie dafür sorgen, dass die Tabellen parallel geladen werden. Zunächst wird **ifddt** die Tabellen mit den richtigen Extent−Größen anlegen und anschließend werden die Daten parallel in die Tabellen übertragen. Auch die Anwendung des HPL stellt dabei kein Widerspruch dar, das heißt die Sub−Optionen des **ifddt** können durchaus zusammen mit den High Performance Loader Optionen angewendet werden.
Zum Aktivieren der parallelen Verarbeitung verwenden Sie die **−g** Optionen. Diese Optionen machen es erforderlich, den Namen der Zieldatenbank anzugeben. Während in früheren Versionen des **ifddt** die Angabe der Zieldatenbank nicht notwendig war, wurde diese Angabe im Rahmen der Parallelisierung notwendig. Sofern Sie eine Kopie einer Datenbank mittels der Option **−n** dbname erstellen, wird als Name der Zieldatenbank der Name nach der Option **−n** verwendet. Wenn Sie jedoch die Daten in ein Verzeichnis entladen, so müssen Sie die Option **−g** dbs=name verwenden.

4.38 Datenbanknamen spezifizieren (−g dbs=*name*)

Mit der Option **−g dbs=name** legen Sie den Namen der Zieldatenbank fest. Dieser kann anschliessend − ausser durch manuelle Änderung in der SQL−Anweisungsdatei − nicht mehr geändert werden. Sofern Sie diese Option weglassen, wird der Name der Zieldatenbank aus dem Namen des Entladeverzeichnisses (−N) oder aus der Angabe nach der Option (−H) ermittelt.

4.39 Autoselect der Parallelverarbeitung (−g ppr)

Mit der Option **−g** ppr weisen Sie **ifddt** an, eine Parallelverarbeitung anzuwenden, wobei **ifddt** die Anzahl der parallel laufenden Prozesse aus der Anzahl der DbSpaces ermittelt.

Besser als die automatische Ermittlung paralleler Prozesse ist die direkte Angabe der parallelen Prozesse:

4.40 Anzahl der parallelen Prozesse festlegen (−g anzahl_prozesse)

Mit der Option **−g** *anzahl_prozesse* wird **ifddt** *anzahl_prozesse* viele Ladeprozesse generieren. Als optimale Einstellung für die Anzahl der Prozesse ergibt sich die Formel:

NUMCPUVPS x 2

wobei Sie den Wert für NUMCPUVPS der Konfigurationsdatei des Zielservers entnehmen. Standardmäßig wird **ifddt** bei der Parallelisierung die PDQ Optionen des Datenbankservers vollends ausnutzen. Bei vielen parallelen Prozessen hat dies aber zur Folge, dass jeder einzelne Ladeprozess relativ wenig Speicher für das Erstellen von Indizes erhält. Gerade wenn große Tabellen geladen werden sollen, würde sich diese Einstellung negativ auf die Geschwindigkeit der Indexerzeugung auswirken. Aus diesem Grund sollten Sie in diesem Fall die PDQ−Priorität für jeden Ladeprozess erhöhen.

4.41 PDQ−Priorität festlegen (−g pdq)

Die Option **−g** pdq_pdqwert legt die PDQ−Priorität fest. Jeder Ladeprozess fordert dabei pdqwert% der PDQ−Ressourcen an und das hat zur Folge, dass bei der Indexanlage für jeden Ladeprozess pdqwert% der Ressourcen zur Verfügung stehen.
Ohne die Option **−g** pdq ergibt sich die Priorität aus:

100% : Anzahl der Ladeprozesse.

Bei vier Prozessen wären das folglich 100% : 4 = 25%. Hätten Sie als maximalen Speicher für die Sortierung 200MB (DS_TOTAL_MEMORY = 200000) eingetragen, so stünde für die Sortierung während der Indexanlage jedem Prozess nur 50MB zur Verfügung. Haben Sie aber Tabellen, deren Indexdaten die 50MB übersteigen, so müsste zur Indexanlage auf temporäre Dateien ausgewichen werden. Würden Sie die Option **−g** pdq=50 angeben, so stünden jedem Ladeprozess 50% der Ressourcen und somit 100MB zur Verfügung.

4.42 Ausgabe für Oracle erzeugen (−g ora)

Beabsichtigen Sie, Ihre Informix Datenbank in eine Oracle Datenbank zu überführen, so können Sie mittels der Option **−g ora** eine SQL Datei für das Oracle Kommando **sqlplus** erzeugen. Sofern Sie nicht nur das Schema, sondern auch die Daten in die Oracle Datenbank übertragen möchten, setzen Sie die Option **−g ora** in Kombination mit der Option **−N** ein. Neben den Entladedateien werden dabei eine Reihe von Kontrolldateien für den *Oracle SQL Loader* erstellt. Der Ladevorgang wird später analog zu der Informix Variante mit dem Aufruf des Kommandos **sqlplus** zentral gesteuert. Innerhalb der generierten SQL Datei befindet sich der allgemein gültige Aufruf des **HOST** Kommandos, mit dem die Daten auf allen gängigen Plattformen geladen werden können.
Vor dem Aufruf des **ifddt** können Sie zur Steuerung des Oracle Loaders diverse Umgebungsvariablen setzen:

Umgebungsvariable	*Beispielwert*	*Bedeutung*
ORA_USER	scott	Benutzername in der Oracle Datenbank
ORA_PASSWD	tiger	Kennwort des Benutzers in der Oracle Datenbank
ORA_IFDDTPATH	C:\\inputpath\\	Pfadname für die Entladedateien im Zielsystem
ORA_LOGPATH	C:\\logs\\	Pfadname der Protokolldateien des SQL Loaders

Es wird dringend empfohlen, die Umgebungsvariablen ORA_IFDDTPATH sowie ORA_LOGPATH zu setzen, da anderenfalls der SQL Loader die Protokolldateien im Oracle Installationsverzeichnis anlegt und die Entladedateien ebenfalls in diesem Verzeichnis sucht!

4.43 inf_storage Datei für BaaN Umgebungen erzeugen

Arbeiten Sie in einer Baan Umgebung, so sollten Sie stets die Steuerdatei namens inf_storage, die sich im Verzeichnis $BSE/lib/informix befindet, pflegen. Einige Werkzeuge der Baan Umgebung benötigen diese Datei, wenn Sie beispielsweise bestehende Tabellen erweitern, reorganisieren oder von einem anderen System Daten übernehmen. In einer Baan Umgebung können Sie stets für jede einzelne Tabelle beschreiben, in welchem DbSpace die Tabelle liegen soll, und mit welchen Extent−Größen die Tabelle ausgestattet ist.
Haben Sie Ihre Datenbank mit Hilfe von **ifddt** reorganisiert, so erstellt **ifddt** auch optional eine Musterdatei für die entsprechenden Tabellen, die Sie in Ihre bestehende inf_storage Datei übernehmen können.
Setzen Sie hierzu vor dem Aufruf von **ifddt** die Umgebungsvariable namens BAANFILES

auf die Mindestgröße einer Tabelle in KiloByte, ab der **ifddt** die entsprechende Tabelle in der generierten infstorage Datei auflisten soll. Ihr Baan Support empfiehlt Ihnen die Nutzung mehrerer inf_storage Dateien; eine für den Normal−Betrieb sowie eine inf_storage Datei für administrative Zwecke, z.B. für das Kopieren von Companies oder zum Erweitern von Tabellen.
Indem Sie die Mindestgröße hochsetzen, können Sie eine kleine infstorage Datei erzeugen, die nur die bedeutenden Tabellen−Definitionen enthält. Setzen Sie die Mindestgröße auf den kleinsten Wert 16, so werden alle Tabellendefinitionen ausgegeben. Nach dem Aufruf erstellt **ifddt** eine Datei namens infstorage im lokalen Arbeitsverzeichnis. Beachten Sie, dass diese Datei keinen Unterstrich enthält. Dies soll Sie vor versehentlichem Überschreiben Ihrer aktuellen inf_storage Datei schützen.
Kopieren Sie die von **ifddt** generierte Datei bitte nicht auf Ihre bestehende inf_storage Datei, sondern übertragen Sie mit Bedacht die von **ifddt** generierten Einträge.

5 Administrative Anforderungen

Die Anwendung mancher **ifddt** Optionen setzen bestimmte Funktionalitäten der beteiligten Datenbankserver und der Hardware voraus. In diesem Abschnitt werden einige Einstellungen näher erläutert.

5.1 Einrichten einer Server−Server−Verbindung

Zum Einrichten einer Server−Server−Verbindung, wie sie zum direkten Kopieren von Datenbanken und Tabellen im Netz notwendig ist, müssen Sie auf beiden beteiligten Server−Instanzen jeweils die sqlhosts Datei so anpassen, dass die jeweiligen Remote−Server−Instanzen dem lokalen System bekannt sind. Das nachfolgende Beispiel zeigt eine typische TCP/IP Verbindung, bei der auf dem Rechner ***munich*** eine Socket− und auf dem Rechner ***berlin*** eine TLI−Schnittstelle verwendet wird.
Neben diesen prinzipiellen Voraussetzungen muß zur problemlosen Übernahme der Daten auch die Benutzerkennung auf beiden beteiligen Rechnern vorhanden sein und als *trusted user* bekannt gemacht werden. Ändern Sie daher − sofern notwendig − die Datei **.rhosts** im jeweiligen Heimatverzeichnis des Benutzers. Tragen Sie dort den Hostnamen des **remote Rechners** ein.
Zum Überprüfen, ob die Verbindung wechselseitig funktioniert, können Sie folgende Abfrage mittels »dbaccess« probieren. Auf der Maschine ***munich***:

```
$ dbaccess sysmaster <<eof
    select count(*) from systables a,
    sysmaster@berlin:systables b where a.tabid = b.tabid
eof
```

Funktioniert der Test fehlerfrei, so können Sie den Test entsprechend auf der anderen Maschine vornehmen. Tritt jedoch ein Fehler auf, so könnte dies möglicherweise an falschen Rechnernamen oder IP−Adressen liegen. Analysieren Sie dazu die begleitende Fehlermeldung und kontaktieren Sie ggf. den technischen IBM Informix Support.

5.2 ONCONFIG−Konfigurationsparameter

Um eine schnelle Index−Generierung zu erzielen, werden bei der Datenübernahme seitens **ifddt** die PDQ−Option aktiviert. Diese können sich nur dann positiv auf die Laufzeit auswirken, wenn Sie entweder in Ihrer Konfigurationsdatei die Parameter :

```
DS_TOTAL_MEMORY
DS_MAX_QUERIES
```

setzen, oder die Werte dieser beiden Parameter on−line mit Hilfe des Kommandos »onmode« setzen. Da die Änderung mittels des Kommandos »onmode« nicht dauerhaft ist, empfehlen wir Ihnen die Verwendung von »onmode«.
Setzen Sie den Wert für **DS_MAX_QUERIES** auf **1** bzw. setzen Sie den Befehl:

```
$ onmode −Q 1
```

ab, damit alle PDQ−Ressourcen für die Datenübernahme zur Verfügung stehen.
Setzen Sie den Wert für **DS_TOTAL_MEMORY** auf $^{2}/_{3}$ des zur Verfügung stehenden virtuellen Speichers, den Sie mit Hilfe des Kommandos:

```
$ onstat −g seg
```

ermitteln können. Wenn Sie 100000KB virtuellen Speicher zur Verfügung haben (100MB), setzen Sie das Kommando:

```
onmode −M 66600
```

ab. Nach einem abschließenden Neustart des Servers werden die mit dem Kommando »onmode« vorgenommenen Änderungen wieder rückgängig gemacht. Sofern die Standardeinstellung der entsprechenden Konfigurationsparameter bereits diese Werte aufweisen, müssen Sie das Kommando »onmode« nicht starten.

5.3 Beschleunigung des Ziel−Servers

Um eine Datenübernahme auf dem Ziel−Server zu beschleunigen, empfehlen wir Ihnen, während der Übernahme in relativ kurzen Zeitabständen einen physikalischen Checkpoint zu erzwingen.
Am einfachsten erreichen Sie dies über die AutoFlush−Funktion, die Sie unter der URL http://www.weideneder.de/download/admin/autoflush.scr herunterladen können und während der Übernahme in einem separaten Fenster laufen lassen.

5.4 Voraussetzungen der Web−Server−Konfiguration

Zum Betrieb des **ifddt** Assistenten, muß sowohl das Programm **ifddt** als auch die

I−CONNECT Schnittstelle auf der gleichen Maschine laufen, wie der Web−Server. Des weiteren müssen folgende Kriterien erfüllt sein:

- Der Web−Server muß unter der informix−Kennung laufen.
- Der Web−Server muß ein UNIX−Shell−Skript ausführen können.
- Die sqlhosts−Datei muß eine Verbindung zu allen beteiligten IBM Informix Dynamic Server Instanzen ermöglichen.

Bei Bedarf können Sie die in der Installationsanleitung beschriebenen Umgebungsvariablen dirket im CGI−Skript an Ihr System anpassen.

5.5 Festplatten−Layout

Bedenken Sie stets, dass beim Entladen von Daten, sowie beim Kopieren von Daten große Datenmengen auf Ihre Festplatten geschrieben werden. Daher ist es sehr empfehlenswert, die Festplatten zuvor auf ihre Geschwindigkeit hin zu überprüfen und ggf. zu optimieren.
Als äußerst nützlich hat es sich bei verschiedenen Tests erwiesen, wenn Sie die Daten auf den Festplatten physikalisch gleichmäßig verteilen. Versuchen Sie probeweise mit Hilfe des UNIX Werkzeugs »dd«, die Geschwindigkeit der Festplatten beim Schreiben großer Datenmengen zu testen, noch bevor sich auf den Festplatten wichtige Daten befinden.
Beachten Sie dabei, dass alle Daten, die sich vor dem Test auf den Festplatten befanden, durch diesen Test zerstört werden!

```
$ time dd if=/dev/zero of=/dev/rchunk1 bs=4k count=100000
```

Mit diesem Test schreiben Sie ca. 400MB auf die Platten, die sich hinter der Gerätedatei ***/dev/rchunk1*** befinden. Die reale Antwortzeit dieses Befehls sollte nicht oberhalb 100 Sekunden liegen. Dauert dieser Vorgang länger als 100 Sekunden, so wenden Sie sich bitte an Ihren Hardware− und an Ihren Betriebssystemhersteller und bitten diesen, die Ursache für die langsame Übertragung auszumachen und zu beseitigen.
Stellen Sie bei dem Test sicher, dass die verwendete Gerätedatei hinreichend groß ist und verändern Sie auch nicht den Test, indem Sie die Datenmenge reduzieren. Kleinere Datenmengen könnten ggf. in einem kleinen Cache−Modul untergebracht werden; ein solcher Test würde dann keinen Rückschluß auf die tatsächliche Dauer eines Kopiervorgangs Ihrer Datenbank zulassen.

6 Bekannte Bugs

Im Zusammenhang mit der Extentgrößen−Ermittung seitens des **ifddt** treten auf verschiedenen Plattformen, insbesondere auf HP−UX und UnixWare, Probleme auf, wenn es sich um den ersten Zugriff auf die **sysmaster** Datenbank handelt.
Innerhalb der generierten SQL−Skripts erscheint am Ende der Tabellendefinitionen dann stets der Hinweis:

```
{ no extent size for partnum = 0x00200026 (1)}
```

Dabei handelt es sich nicht um ein Problem seitens **ifddt**, sondern um ein Problem im Zusammenhang mit dem IBM Informix Dynamic Server. Sie können dieses Problem umgehen, indem Sie folgenden Befehl auf der Kommandozeile absetzen:

```
$ dbaccess sysmaster <<eof
select tabname[1,18], nrows from systabnames a,
sysptnhdr b where a.partnum=b.partnum and dbsname = 'dbname';
eof
```

Ersetzen Sie dabei *dbname* durch Ihren Datenbanknamen und starten Sie diesen Befehl solange, bis alle ihre Tabellennamen auf der Ausgabe erscheinen. Anschliessend können Sie **ifddt** erneut wie gewünscht starten. Prüfen Sie aber die Ausgabe nochmals hinsichtlich der Meldung *»no extent size«*.